Politecast – a new communication primitive for wireless sensor networks

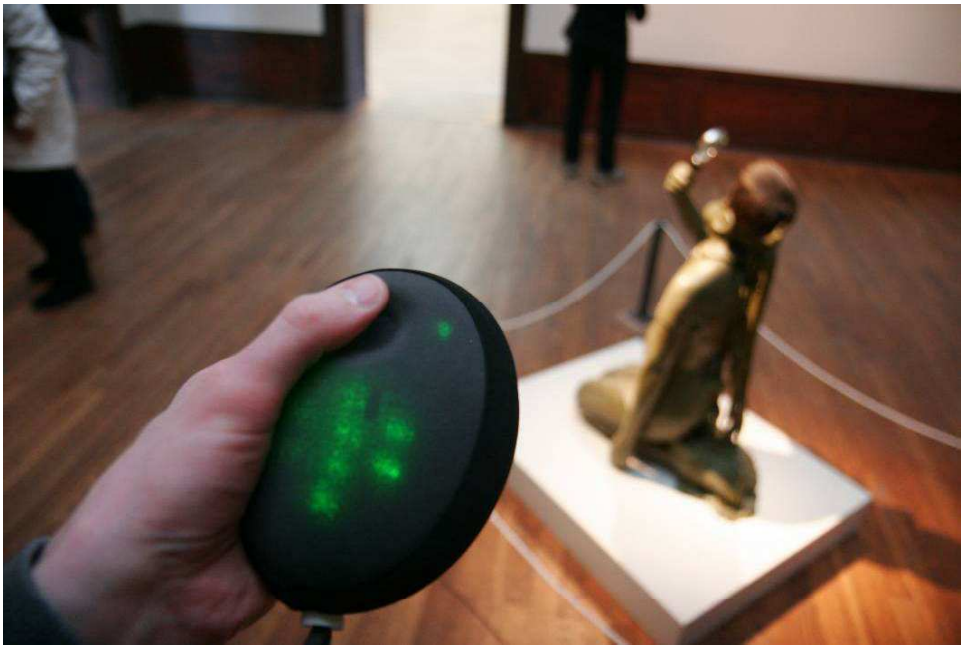
MARCUS LUNDÉN

KTH Industriell teknik
och management

Master of Science Thesis
Stockholm, Sweden 2010

Politecast – a new communication primitive for wireless sensor networks

Marcus Lundén



Master of Science Thesis MMK 2010:80 MDA 393
KTH Industrial Engineering and Management
Machine Design
SE-100 44 STOCKHOLM

KTH Industriell teknik och management

Examensarbete MMK 2010:80 MDA 393

Politecast – ett nytt kommunikationsprimitiv för trådlösa sensornätverk

Marcus Lundén

Godkänt	Examinator	Handledare
2010-09-29	Mats Hanson	Mikael Hellgren
	Uppdragsgivare Swedish institute of Computer Science (SICS)	Kontaktperson Adam Dunkels

Sammanfattning

Trådlösa sensornätverk är en potentiell framtida jättemarknad. Ericsson förutspådde tidigare att 50 miljarder enheter kommer vara uppkopplade mot Internet år 2020. En stor andel av dem kommer att bestå av trådlösa sensornätverk. Innan det kan bli verklighet finns det en del problem att lösa för att enheter ska kunna genomleva åratal av kontinuerlig drift utan att behöva byta kraftkälla. Dessa enheter är typiskt små, billiga och har begränsade prestanda i bandbredd, minne och CPU. Kommunikation kommer huvudsakligen ske trådlöst, och radiodelen är typiskt den mest kraftslukande komponenten. Att reducera radioanvändandet är därför det viktigaste sättet att öka livslängden.

I denna rapport identifierar jag fyra problem med det vanliga broadcast primitivet. Jag implementerar ett nytt kommunikationsprimitiv kallat Politecast. Politecast utvärderas i tre fallstudier: en enkel applikation kallad Steal the Light, en simulering av en Neighbor Discovery-applikation samt en två månaders installation i en konsthall: Lega-systemet. I dessa visades politecast kunna ge längre livstid, mindre mängd radiotrafik samt högre prestanda i applikationerna.

KTH Industrial Engineering and Management

Master of Science Thesis MMK 2010:80 MDA 393

Politecast – a new communication primitive for wireless sensor networks

Marcus Lundén

Approved	Examiner	Supervisor
2010-09-29	Mats Hanson	Mikael Hellgren
	Commissioner	Contact person
	Swedish institute of Computer Science (SICS)	Adam Dunkels

Abstract

Wireless sensor networks have the potential for becoming a huge market. Ericsson predicts 50 billion devices interconnected to the Internet by the year 2020. Before that, the devices must be made to be able to withstand years of usage without having to change power source as that would be too costly. These devices are typically small, inexpensive and severally resource constrained. Communication is mainly wireless, and the wireless transceiver on the node is typically the most power hungry component. Therefore, reducing the usage of radio is key to long lifetime.

In this thesis I identify four problems with the conventional broadcast primitive. Based on those problems, I implement a new communication primitive. This primitive is called Politecast. I evaluate politecast in three case studies: the Steal the Light toy example, a Neighbor Discovery simulation and a full two-month deployment of the Lega system in the art gallery Liljevalchs. With the evaluations, Politecast is shown to be able to massively reduce the amount of traffic being transmitted and thus reducing congestion and increasing application performance. It also prolongs node lifetime by reducing the overhearing by waking up neighbors.

Acknowledgments

I would like to express my cordial gratitude to my supervisor Adam Dunkels for not only giving me invaluable feedback but also guidance and inspiration (not to mention patience with some not-so-clever-questions). I am also utmost thankful for the pleasure to share blessings and cursings with the team behind the Lega, of which Jarmo Laaksolahti, Jordi Solsona, Jesper Karlsson, Helena Mentis, Anna Karlsson, Petra Sundström, and many more, have made the darkest hours of debugging much, much easier to handle. Many deep thanks to the Networked Embedded Systems group at SICS for not only support but being excellent colleagues. Special thanks also go out to the MobileLife group for many interesting discussions, and the enforced — but always appreciated — socializing at the Wednesday fika. I thank my supervisors and examiners at KTH for feedback, support and patience when I was more focused on improving performance of device drivers rather than actually writing the thesis: Bengt Eriksson, Mats Hanson, Mikael Hellgren. For my closest friends and family, I extend my most sincere, deep and warm gratitude for you being who you are and your firm support.

This thesis has been pursued within the Designing Systems for Supple Interaction project, which is a collaboration with SICS, The Mobile Life Center, Wireless@KTH, SonyEricsson and Ericsson, and funded by SSF. Adam Dunkels is the originator of the Politecast principle.

Contents

List of Figures

List of Tables

Code listings

Chapter 1

Introduction

Wireless sensor networks have the potential to become a huge market and fulfilling the old dream of ubiquituous computing but before that can happen, there are some problems to solve. The power available to a node is assumed to be small so the node must be power efficient. The radio is often the most power consuming device on a node, and as such it is common to switch it off as much as possible – a tradeoff between power and performance. A common problem is one-to-many communications, which has inherent problems associated with it. In this thesis, four such problems are identified and politecast is presented and evaluated as a way of dealing with those problems.

1.1 Wireless Sensor Networks

Wireless sensor networks (WSN) consist of many nodes with wireless communication and sensors for e.g. temperature, humidity, movement, light or magnetic fields. Applications range from industrial monitoring and control, habitat and nature monitoring and security applications to consumer products such as health monitoring or games and lifestyle devices in PANs (Personal Area Networks). The technology of WSNs is predicted to be a huge market. Oil company BP has with WSNs costs 1/20 that of conventional sensors [15], HP predicts billions of sensor nodes on buildings, along roads etc [13]. Ericsson ups this number to 50 billion devices connected to the Internet by the year 2020 **??**, many of which are going to be WSNs. In this envisioned world with ever increasing number of objects with computing capabilities all around us, wireless communication and long lifetime is key. Wirebound communication is cumbersome and expensive, reaching costs of on the order of 10–1000 USD/foot [17].

1.2 Problem Formulation

With many devices, perhaps in remote locations, long life span and the ability to communicate are crucial. In low power wireless sensor networks it is often

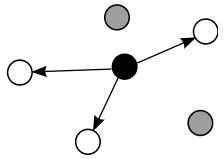


Figure 1.1: Politecast principle — transmit without waking up nodes so that only awake and interested nodes (white) are bothered. Grey nodes are disinterested and sleeping.

desirable or unavoidable to communicate one to many, but this is problematic as it consumes a lot of power. It is not feasible to recharge or change batteries too often. If equipped with energy harvesting technology, the power available will be small, on the order of ca 0.1–0.2 W when equipped with solar panels the approximate size of a mote. Therefore, minimizing power consumption is crucial. Overheard transmissions are power wastes when a receiver is disinterested in the information. How can the design space for the application designer be extended with a better mechanism for sending from one to many, where disinterested nodes are affected as little as possible?

1.3 Hypothesis

The basic principle behind efficient communication is that the receiver is interested in the data being communicated. The present broadcast primitive assumes that all nodes within range are interested and should receive all one-to-many transmissions. A better approach in some applications or networks could be the opposite — assuming instead that most nodes are disinterested and those that are interested make sure they listen for the transmission.

By shifting the bulk part of the effort to the receiver, the sender can save more energy. This could be especially well suited in situations where many nodes transmit data but few actually are interested, such as radio beacons and periodic neighbor discovery announcements. As no time or transmissions are spent on waking up, latency and congestion can be lower. The power consumption might be higher for the receivers if they listen often or for long durations, but it gives the application designer an added flexibility as interest (i.e. listening) could be triggered e.g. by movement.

1.4 Method

This thesis has been conducted in an explorative and experimental way. The qualitative properties of the primitive was constantly under scrutiny as I developed the API and implementation. The quantitative properties were evaluated in simulations and experiments by using it in applications and comparing it with the performance of the same applications but using broadcast. The necessary

software components needed were built as the need occurred. Examples include device drivers for external hardware, a position data generator and software for parsing message logs and calculating statistics.

1.5 Thesis Structure

This section briefly described the context and the thesis. Section 2 gives insight to the background and the problem domain. Sections 3 and 4 start with identifying four problems with broadcast, then present the properties of politecast and how it was implemented in Contiki. Then, the three case studies used for evaluation and the evaluation itself are described in Section 5. Section 6 elaborates on related work. The thesis follows with a discussion of the results together with possible areas of future work and a conclusion in Section 7. Appendices are API references (App. A), code examples (App. B), description of the developing environment (App. C), glossary (App. D) and references (App. D).

Chapter 2

Background

Wireless sensor networks have been around for decades but is still awaiting wide industrial adoption. There have been deployments for academic purposes, such as a system for sharing feelings among coworkers, a volcano monitoring system and a road tunnel monitoring system.

2.1 Wireless Sensor Networks

Today, wireless sensor network technology is pacing up to become the anticipated ubiquitous computing reality. It has a long history, going back to the early days of computing and integrated electronics. One of the first wireless sensor networks was deployed from the air over Vietnam during the war in the 1970's – the ADSID, or Air Delivered Seismic Intrusion Detector. American troops wanted to be able to track enemy troop movements on key roads in the jungle and dropped sensor nodes that resembled darts into the ground. They had seismic sensors, large batteries and a radio transmitting to airplanes circling above. They weighed circa 20 kg each, measuring circa 1,2 meters tall and could last for a few weeks. Present technology makes use of inexpensive commercial off the shelf (COTS) products, are small and power efficient and come with a variety of sensing abilities. It is an emerging market with huge potential. They can today be seen mostly in academic and industrial applications. Figure 2.1 shows a future low-rate wireless personal area network formed by the devices and clothes worn on a person.

2.1.1 Examples

Several deployments have shown the strengths and weaknesses together with challenges of WSNs.

FriendSense FriendSense [50] was a system for exploring how co-workers in an office space could express feelings and wishes with other co-workers. Every

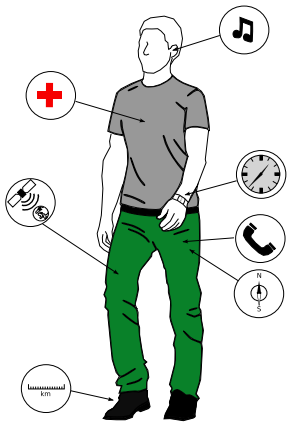


Figure 2.1: An LR-WPAN where the mobile phone connects wirelessly with health monitoring devices in the clothes, providing up to date status, and with wireless headphones for streaming speech and music. Sensors for distance and location tracking are present for safety and statistics. Every device is expected to work seamlessly and without nuisances as a supple system.

participant received a personal node that used accelerometer, temperature and humidity sensors to shape an avatar on a large public screen. One node acted as data sink and relayed data to the server application running the screen.

ZebraNet ZebraNet [36] was meant to be an aid for wildlife scientists doing research on the movement patterns of zebras. Every zebra in the pack under study had a node with a GPS-receiver, solar power panel and sensor node around its neck, which opportunistically sent the gathered data towards the data sink.

Great Duck Island The deployment at Great Duck Island [52] was conducted to provide wildlife scientists with habitat data (temperature etc). It consisted of circa 150 small nodes buried in the nests of wild birds on the island for four months together with a number of weather sensor nodes placed on the surface. In all, the deployment was a great learning experience for the wireless sensor network community as it showed the many difficulties and differences with real life deployments in contrast to testing in labs.

Volcano monitoring On the potentially more dangerous side, a volcano monitoring sensor network [56] was deployed in Ecuador for 19 days. Because of the cost and size of regular volcano monitoring equipment, a usual volcano monitoring deployment only have a couple of sensor sets, rendering course grained observation data. With sensor nodes, the scientists got data from 16 nodes scattered over the surface. This deployment showed how important it is with ground

Microcontroller	Texas Instruments MSP430F1611 [5]
Radio transceiver	Chipcon CC2420 [4]
External flash memory	ST M25P80 [3] 1 MByte
Antenna	Antenova Impexa 2.4 GHz [1]
Accelerometer	Freescale MMA7260 3-axis multiple sensitivity [2]
LED	8 red SMD LEDs

Table 2.1: The hardware components in Sentilla JCreate nodes.

truth and calibration of sensors as the data collected was of subpar quality to the volcano scientists.

Related work and more examples are described in Section 6.

2.1.2 Examples of Hardware

Most of the motes on the market consists of a power source, a microcontroller, a radio transceiver, an antenna, a flash memory for data storage and sensors according to the application at hand. Common properties are that they are small, resource constrained and inexpensive (on the order of 50-100 USD).

Sentilla Jcreate Nodes JCreate nodes from Sentilla [11] have a microcontroller and a radio transceiver from Texas Instruments. The microcontroller is a 16-bit microcontroller with 10 kB RAM, 48 kB flash, 12-bit ADC, hardware multiplier and DMA. The radio works in the free ISM-band at 2.4 GHz and is compatible with IEEE 802.15.4. The power supply is two AAA-sized 1.5 V batteries in a battery holder on the back of the node. Eight LEDs, a flash memory and a low-g, three-axis accelerometer are also included in the node. It has a chip antenna from Antenova. Nodes are programmed using a programming fixture which connects via USB to a computer.

Tmote Sky Nodes Tmote Sky nodes use the same radio, flash memory and microcontroller as JCreate. They have onboard USB circuitry and different sensors, such as humidity and infrared light. For power source they use two AA-batteries.

2.1.3 Characteristics

A WSN consists of many nodes (tens to thousands), small devices also called motes, that are severely resource constrained in comparison to e.g. industrial embedded systems for motor control. There is often a coordinator or master node with less constraints; for instance it can be connected to the Internet and have constant power supply. That master can act as a data sink.

Power Consumption Contrary to what one could expect, many of the common radio transceivers found in 802.15.4-compliant nodes actually need equal or more power listening than transmitting [30]. This is because of that the radio transceiver needs to have amplifier, filter and logic active, and most 802.15.4-devices has a low maximum transmission power (on the order of 1 mW). E.g. the CC2420 radio transceiver consumes circa 57 mW when listening, but only 51 mW when transmitting at full power. This means that the radio transceiver uses about $\frac{P_{CC2420}-P_{Tx}}{P_{Tx}} = \frac{51-1}{1} = 50$ times more power on logic etc. than on actual transmitted signal power. For comparison, the accelerometer [2] and microcontroller [5] in the Sentilla JCreate consumes circa 1.5 mW and 3 mW respectively when active.

Communication Communication in low power wireless networks is counter-intuitive in terms of connectivity, consistency over short time and probability of reception. Figure 2.2 is a graph of the momentary probability of a successful transmission. It was compiled by the authors of [33] by having a grid of nodes on an open field ($15 * 15\ m^2$) sending over 50000 packets from a node in the center while the surrounding nodes kept track of how many were received. Wireless transmissions are not static, uniform discs of radiation. They are often asymmetric [33], i.e. node A hears node B but B cannot hear A. In [33], the number of observed asymmetric links were between 5 – 15 % of the whole depending on distance and transmission power. To make things worse, this constantly changes over time and show burstiness [48]. Burstiness is that packet losses are not independent from each other, but correlated with the number of failed or succeeded immediately previous transmissions. The reason for this is unclear, but a hypothesis in [48] is interference from 802.11b wireless networks that operate in the same frequency band and has on the order of 100 times larger transmission power. Other sources of these behaviors are interference from microwave ovens and switching power supplies, multipath reflections, congestion, antennas not being omnidirectional etc.

The hidden terminal problem can cause serious performance degradation in dense networks. Before a transmission, the node must check the radio media for traffic by sampling the channel for energy – clear channel assessment (CCA). If the energy is below a threshold it starts transmitting. If two nodes who are out of range of each other want to send to the same receiver (as seen in Figure 2.3), one can sample the radio and not sense the ongoing transmission, thus starting its own transmission and causing interference at the receiver.

Also, multi-path phenomenas are common. Radio waves are reflected on surfaces and arrives at the receiver multiple times with slight time skew. Other problems include overhearing of uninteresting transmissions and congestion in the radio medium when there is a lot of traffic.

2.1.4 Challenges

Probably the most important challenges with wireless sensor networks – considering the predicted future of many billions of devices – are lifetime, scalability

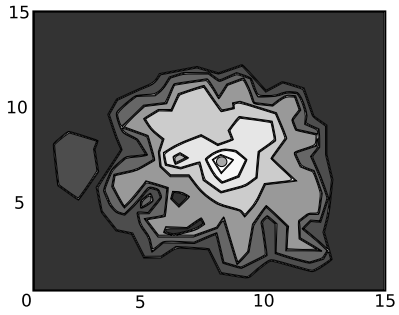


Figure 2.2: The graph (after empirical measurements in [33]) illustrates the momentary probability of a successful transmission if a receiver was placed in the vicinity of the node on the $15 * 15$ m^2 field. Darker means a lower probability. This shows the non-uniform and asymmetric nature of wireless communications. Notice the islands of worse or better probability, in part explaining asymmetric links.

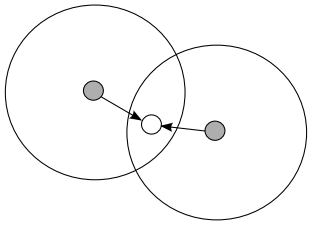


Figure 2.3: The hidden terminal problem: two nodes that want to transmit data to the same node at the same time samples the radio medium for energy (CCA). As they are out of range, they detect none and start transmitting, resulting in interference at the intended receiver.

and mobility. Other challenges are security, reliability, predictability and network topology issues such as routing.

Lifetime A critical factor for WSNs is lifetime. The sheer numbers of nodes and the sometimes remote location of where the network is deployed [36] [52] [53] [56] makes it difficult if not infeasible to access the nodes physically. Energy harvesting, such as solar, kinetic or wind power, is a hot research topic. However, because of their low efficiency (commercially available solar panels have at best 20 % efficiency [16]) and increasing demands on applications, low power consumption will still be necessary. Also, depending on the application, a node can spend far more time listening for transmissions rather than transmitting or receiving. And, as listening cost more than transmitting, just sending less is not enough.

Scalability WSNs are predicted to include huge numbers of nodes and with increasing density comes an increasing amount of traffic a node can hear. Many of the current state-of-the-art power saving MAC protocols are designed for WSNs with a relatively low number of nodes. If several nodes transmit at the same time, congestion and collisions will occur, leading to corrupt and dropped packets and makes it hard to get a message through. If the network use a synchronous MAC protocol, it can run out of timeslots. Memory can be an issue if nodes need to keep neighbor tables and they are too many.

Mobility As WSNs become more common, and especially in LR-WPANs, they are expected to also be more mobile. Motion is unpredictable, especially when people or animals are involved. Devices moving independently also implies a dynamic structure of the network itself, as devices move in and out of range of other devices. This again points to the problem with lifetime as a more dynamic network implies more traffic.

For a network to be able to form, so devices are able to share information, they must discover each other. This process is called neighbor discovery and should be a fundamental and constantly ongoing process for every device [29]. However, this is costly as a node must periodically advertise its own presence and listen to others. Neighbor discovery should be accurate (i.e. not missing neighbors), power efficient and have low latency. For a mobile WSN, it should be asynchronous.

2.2 The OSI Reference Model

The OSI reference model is an abstraction for layered communication and protocol design, divided into seven layers. The layers can pad the data with extra information, like CRC-checksums, address fields and flags. When received, the corresponding layer strips the packet of this information, processes it (e.g. by checking CRC) and pass on to higher layers. Figure 2.4 illustrates this as a packet is sent down the protocol stack, over the physical layer to the receiver and up through the layers.

For this thesis, only the bottom layers are important. IEEE 802.15.4 only specifies the PHY- and MAC layers, and the politecast communication primitive is logically located in the data link layer.

NETWORK-layer The network layer uses logical addresses to direct communication to nodes. In IP, the addresses are 32 bits (IPv4) or 128 bits long (IPv6). In Rime, the addresses are 16 bits.

DATA LINK-layer The data link layer provides means to communicate with other devices by physical addresses (the MAC address). It can be divided into two sublayers: media access control (MAC) and logical link control (LLC).

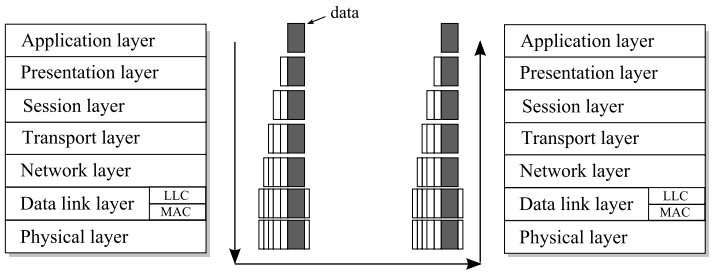


Figure 2.4: The seven layer OSI model. The data generated in the application at the sender (left stack) is padded with information (e.g. CRC-checksum, addresses, application version) at some layers so that they can accomplish their respective tasks at the corresponding layer at the receiver (right stack), such as error handling, managing communication flow etc.

MAC controls how and when the LLC-layer can access the media for transmitting and receiving and LLC wraps the packet with information for e.g. error correction.

PHY-layer The bottom layer specifies the electrical and physical properties: the voltage levels, modulation scheme, power, frequency, mechanical connectors etc.

2.2.1 Power Saving MAC Protocols

For nodes such as Sentilla JCreate and Tmote Sky, the radio transceiver consumes approximately ten times more power than the rest of the sensor node together (if no LEDs are on). For example, the energy used for transmitting 1 byte of information with the CC2420 roughly equals 2200 clock cycles with the microcontroller at 4 MHz. This is common for sensor nodes as other components often are power efficient in comparison to the radio. The MAC layer protocol is handling switching on or off the radio to save energy. MAC protocols can in large be divided into three subsets: asynchronous, synchronous and hybrids.

Asynchronous MAC Protocols Asynchronous MAC protocols are not synchronized in time and as such must incorporate a mechanism for either making sure the receiver is listening, or wait for it to tell that it is listening. They can in large be divided into LPL and LPP schemes, where two well-known examples are X-MAC and RI-MAC respectively.

X-MAC X-MAC [21] was presented in 2006. It is a low power listening scheme (LPL) in which the nodes periodically wakes up and listens for traffic a short time before going back to sleep.

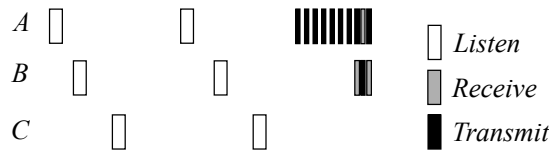


Figure 2.5: A sending an unicast packet with X-MAC to B. As soon as B hears the strobes, it responds with an ACK, and A sends the data packet.

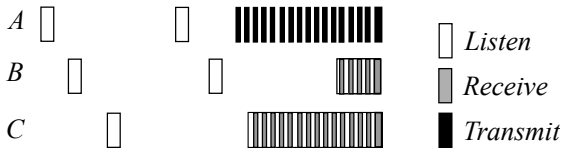


Figure 2.6: Sending a broadcast packet with X-MAC. The strobe train duration is slightly longer than the sleeping period.

In X-MAC, nodes periodically switch their radio on to listen a short while for transmissions. A sender must make sure that it transmits during this slot, but as it does not know when that slot occurs, it must repeatedly transmit wakeup strobes until the receiver responds. For a broadcast, the wake up strobes must be transmitted for more than the globally defined sleeping period T.

Four parameters shape the timings of X-MAC, for example how many strobes are transmitted, see Figure 2.7.

ContikiMAC ContikiMAC is a MAC protocol that comes with Contiki. It is similar to X-MAC but instead of a continuous listening in each wake up period it instead does two CCA checks with a small delay between, during which it sleeps. This makes for a lower radio utilization. Transmissions do not use strobes as X-MAC but instead transmit the actual data packet repeatedly (Figure 2.8). ContikiMAC timings are set by specifying a channel check rate, such as 16 Hz.

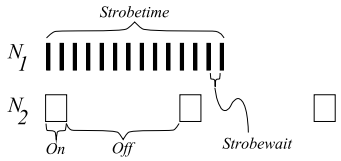


Figure 2.7: The figure shows the four parameters that define the most basic properties of X-MAC. As can be seen from the figure, the Strobe wait time must be shorter than the On time. Also, the Strobe time must be longer than the sum of On and Off times.

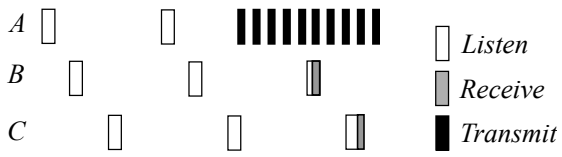


Figure 2.8: With ContikiMAC, the sender (node A) repeatedly transmits the data packet itself. Any node in range waking up will receive the packet and go back to sleep.

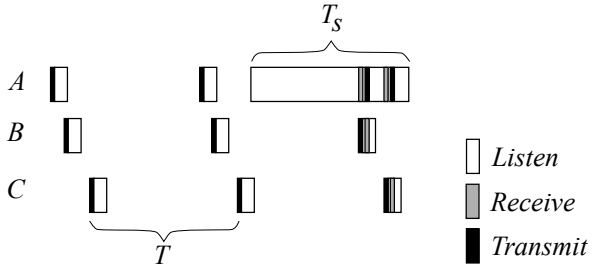


Figure 2.9: RI-MAC sending a broadcast packet. The sender is awake for longer than a sleeping period and responds to hearing beacons from each waking node by sending the data packet.

RI-MAC RI-MAC [49] was proposed in 2008 as a response to the scalability and congestion problems with X-MAC and other LPL-protocols. It is a low power probing scheme (LPP) in which the nodes wake up and probe the others for data by sending a beacon, which nodes can respond to if they have a packet pending for that receiver. When broadcasting, the sender lies awake and awaits probes for an entire sleeping period (Figure 2.9).

Synchronous MAC protocols Synchronous MAC protocols divides time into slots (TDMA, Time Division Multiple Access), which are assigned to the nodes. If node A wants to transmit to node B, it must await the time slot when it knows B is listening. For a broadcast, it must lie awake and transmit at every slot.

WSNs are envisioned to be used in large numbers with inexpensive hardware. Such hardware will use inexpensive oscillators (or even the internal oscillator in the microcontroller) which are of lower quality and precision. This means that the clock drift will be high. The oscillator drifts due to impurities in the crystal, temperature, age etc and is expressed in ppm. Regular oscillators used in sensor nodes often have between 20–100 ppm drift, which corresponds to a drift of ±80–400 clock ticks per second for a Sentilla JCreate (at 4 MHz).

Because clocks need to be synchronized with high accuracy, and the oscillators in the nodes drift, synchronization messages need to be sent. For an

accuracy of 1 ms with a 20 ppm oscillator, a resynchronization message is needed every 50 seconds [37]. This synchronization overhead can easily be dominant, depending on sleep periods and how often transmissions of useful data occurs. In the Great Duck Island second deployment [52], nodes sampled the sensors every 20 minutes and transmitted this to the data sink node. This would mean, with a 20 ppm oscillator, the number of synchronization messages would outnumber the sensor messages with 24 to 1.

S-MAC S-MAC [60] uses synchronization messages to exchange schedules with neighbors. It forms microclusters to set up a common sleeping schedule. The sleep and awake periods are predetermined and constant. The main disadvantages with S-MAC are the overhead from the synchronization messages and the periods being constant, which lowers the overall throughput.

Hybrid MAC protocols Hybrid MAC protocols incorporate functionality of both asynchronous and synchronous protocols, or can switch between modes.

MH-MAC MH-MAC [20] can be in either of three states: fully on, asynchronous or synchronous. It uses a RTS/CTS exchange sequence when in full or synchronous mode. When in asynchronous mode, it is similar to X-MAC with preambles.

WiseMAC WiseMAC [31] uses synchronous functionality for optimizing asynchronous traffic. By keeping a table over the neighbors sleeping schedules, it can dynamically adjust the preamble length to start just before it thinks the neighbor will wake up. Broadcast is implemented by buffering the packet and awaiting the anticipated wake up from its neighbors, then transmitting the packet for each neighbor.

2.3 IEEE 802.15.4

The standard IEEE 802.15.4 [14] (hereafter just "802.15.4") was first specified 2003. The focus for the standard was wireless communication between devices in a LR-WPAN. The devices are supposed to have low cost, low power, low complexity, low data rates and long expected lifetime. The standard specifies the two lowest layers in the OSI-model: the physical- (PHY) and the media access control (MAC) layers.

Physical Layer

Devices operate in three open and unlicensed frequency bands: 868–868.8 MHz (3 channels), 902–928 MHz (30 channels) and 2.400–2.4835 GHz (16 channels, 11 to 26) as can be seen in Figure 2.10. The modulation is DSSS/O-QPSK and in the 2.4 GHz band the data rate is a modest 250 kbit/s (lower rates in the other bands). The radio must have a receiving sensitivity of -85 dBm or better

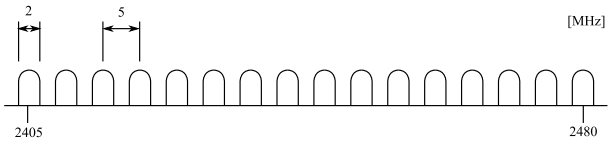


Figure 2.10: The 802.15.4 2.4 GHz radio band. Each of the 16 channels uses 2 MHz and is 5 MHz apart center to center.

(i.e. lower). For transmitting, a minimum maximum transmission power of -3 dBm is required. This means that by specification, as 0 dBm equals 1 mW, a transceiver should be able to sense at least 5 pW and transmit at least at 0.5 mW. Further, a maximum maximum transmitting power is not regulated. Instead, the local regulations where the devices will be used must be followed. Most often, it varies from 1 mW to 1 W depending on location, duty cycle and modulation scheme.

Radio Metrics Two important metrics are link quality indicator (LQI) and received signal strength indicator (RSSI). For every packet the radio transceiver receives, it measures the bit error rate and calculates the LQI. The transceiver can also directly measure the signal strength. RSSI is not a part of 802.15.4 but commonly implemented in the radio ICs on the market anyway. It is measured in dBm.

MAC Layer

802.15.4 does not specify the entire datalink layer, only the MAC layer. The MAC features e.g. CSMA, beacon management, channel access, frame validation, frame format and acknowledgments. Beacons are synchronous and used e.g. for network discovery services. The maximum packet (PDU) size including all overhead is 128 bytes.

2.4 Related Specifications and Standards

2.4.1 Zigbee

Zigbee is a proprietary protocol specification that builds on 802.15.4. It forms networks in either beacon or non-beacon mode. In non-beacon mode, the node must keep the radio on constantly as it uses an unslotted CSMA-CA access. In beacon mode, the node periodically adverts its presence with synchronous beacons. The duty cycle is chosen depending on data rate, but for 250 kbit/s the valid range is 15 ms to 252 s. It has received critique against a too centralized network structure where a network coordinator node can run out memory or not be able to give a child device a network address if it has many neighbors **??**.

2.4.2 Bluetooth

Bluetooth is specified in IEEE 802.15.1. It was designed not for being ultra power conserving but for multimedia appliances and thus have a different application domain than 802.15.4 devices. It uses the same frequency band at 2.4 GHz (but with a different modulation scheme: frequency hopping spread spectrum, FHSS) and has data rates up to 1 Mbit/s (v1.2) or 3 Mbit/s (v2.0). It forms piconets with one master and up to seven slaves and the nodes must remain active in order to be polled whenever the master does that. There has recently been movements towards a broadened, specification of low power Bluetooth with similar key benefits as IEEE 802.15.4.

2.4.3 IEEE 802.11, WiFi

IEEE 802.11 consists of several standards, but those most people know of are b, g and n. They are found in products like desktop and laptop computers, handheld devices etc. The demands on power and data rates are much higher than that of 802.15.4. They use different frequency bands, where b, g, and n are centered around 2.4 GHz and has 14 channels of 22 MHz each. As it uses the same frequencies as 802.15.4, there is a potential risk of interference. The IEEE 802.15 TG4 concluded that in 802.11 and 802.15.4 coexistence there is a risk of performance degradation, but in practice, low duty cycle applications and CCA will help reduce the severity of the interference. The term "WiFi" is a trademark owned by the WiFi alliance for devices that are certified by the WiFi alliance. The requirements for certification are very close to, if not the same as, the IEEE 802.11 standards.

2.5 Network Primitives

A primitive is the simplest or lowest element available for achieving something in programming. It is a term relative to level of discussion because of its generality. Network primitives are used as abstractions of communication, reducing complexity for the application designer by hiding the underlying communication and functionality under the hood. Each primitive represents a way to address and transmit data.

Unicast When node A communicates with a single other node, B, by addressing it, this is said to be an unicast. No other node should take notice of that packet.

Broadcast A broadcast is sent from one node to all nodes. It is commonly used for tasks like neighbor discovery, data dissemination and route discovery. Broadcast stands out as the only primitive usable when the receiver addresses are unknown.

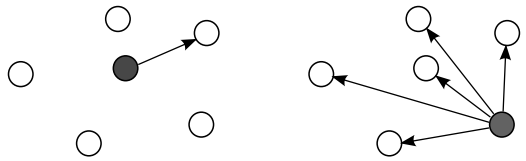


Figure 2.11: Unicast (*left*) and broadcast (*right*) principle.

Multicast Multicast is when a node sends to many nodes. E.g. node A sends a packet to nodes B, C and D, but not to F and G. Addressing can be done with group addresses.

Reliable Unicast Reliable unicast uses acknowledgment packets (ACK) after a successful transmission. If the sender does not receive the ACK within a time period, it retransmits the packet until it either receives an ACK or it has tried too many times.

Flooding Flooding can be used for route discovery (enabling multihop unicast etc) and data dissemination. If node A wants to know the route to node E, it broadcasts a packet with information of the route final destination. As nodes receive the route request message, they resend it. When node E receives the route discovery request, it sends back an ACK via unicast to the node that last sent the route request. That way, the ACK finds its way back to A and A can read the accumulated addresses of the nodes the packet passed by.

Multihop Unicast When a node wants to send to another node that is out of range, it can send via intermediate nodes that relays the packet. This requires the knowledge of a route to the end receiver. E.g. node A wants to send a packet to node E but is out of range. It knows that it can send via nodes B, C and D to E. The relay nodes read the packet headers and resends the packet so it can travel further down the chain to the end receiver.

2.6 Contiki Operating System

The Contiki operating system [18] is a light weight operating system for resource constrained networked devices. It is written almost entirely in the C programming language. This makes it highly portable and easy to learn as it does not require learning any special tools or programming language. The entire Contiki is open source and available for free in a BSD-style license. A key property with Contiki is to be lightweight in order to be used in platforms with extremely scarce resources in terms of energy, computing power and memory. A typical deployment with Tmote Sky-nodes using the Contiki OS and applications is on the order of 2 kbyte RAM and 20 kbyte ROM. Contiki provides many features such as dynamic loading of application modules, enabling over the air (OTA)

reprogramming of nodes in deployment. A miniature TCP/IP-stack [24] enables TCP/IP-communication.

Protothreads By using protothreads [25] the OS achieves multithreading behavior on top of an event-driven kernel. The big advantage with protothreads compared with other ways of implementing multithreading is that the protothreads share the same stack, in contrast to having a complete stack for each thread no matter how much of it is used. This results in a much lower RAM footprint. It does on the other hand introduce minor complications, such as local variables not being saved across a blocking wait (use static variables instead) and that switch cases cannot be used inside a protothread. The overhead is small, only two bytes per protothread.

Processes and Program Flow Processes are implemented as protothreads. They can be polled (by the kernel) or called upon by other processes, with events. When called upon, they run till completion or a blocking wait; they cannot preempt each other. As processes are non-preemptive, care must be taken so that a process will not cause CPU starvation, which results in a reboot due to watchdog timeout.

Clocks and Events Two hardware timers in the microcontroller are used to provide Contiki with a coarse- and a fine-grained clock. Macros provide an easy way of using time in applications. The kernel has an event scheduler that dispatches events to running processes. Events can be asynchronous, synchronous or polling. The two first differ in when the event receiver process is scheduled and polling is done periodically by the kernel, or from within the processes themselves (typically hardware close processes, like device drivers) [18].

Power Saving and Energy Estimation Microcontrollers generally offer low power modes (LPM) where internal components (like ADCs) are shut down to save energy. The Contiki event scheduler runs and polls the scheduled processes so that it can put the microcontroller in LPM until the hardware timer interrupt (or another interrupt) strikes. This way, the microcontroller is in LPM as soon as no process is currently using the CPU. As lifetime is important, it follows that knowing about power consumption is important. Added hardware (such as SPOT [35]) increases size, cost, power consumption and complexity. Energest [27] is a Contiki module for estimating the energy used by a peripheral. Energest uses the fine-grained software timer for measuring the time spent between macro calls. A driver for a peripheral can call an energest macro on entry and just before exit and the energest module periodically sums the time.

Communication Architecture

The communication architecture in Contiki uses a single buffer for both incoming and outgoing packets: the packet buffer. The complete communication stack

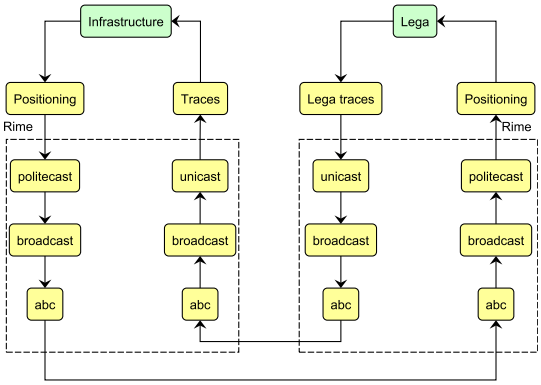


Figure 2.12: The layered structure of the Rime communication primitives; both politecast and unicast builds on the broadcast primitive, adding meta data fields. The politecast connection is used for the positioning service, and when a trace is transmitted from the Lega to the infrastructure, unicast is used.

consists of a high level protocol or application, the Rime collection of communication primitives and the packet buffer, the Chameleon layer [26] with header transformation modules, the MAC layer protocol and finally the device drivers.

A high level protocol writes data into the packet buffer and calls one of the Rime communication primitives, e.g. reliable unicast. This sets the corresponding attributes (meta data fields), and calls Chameleon. Chameleon uses the appropriate module to create headers (for e.g. 802.15.4 or IP) according to the attributes. This forms the packet data unit (PDU), which is forwarded to the MAC protocol, which uses the device drivers to transmit it. With Chameleon, the high level protocols and Rime primitives do not need to be changed e.g. when moving from 802.15.4 to Ethernet.

Rime The communication primitives in Rime form a layered structure, where the more complicated primitives builds on one or more of the more basic primitives. The most simple primitive is anonymous broadcast ("abc"). The broadcast primitive adds sender address and pass the packet to abc. The unicast primitive adds the receiver address attribute to broadcast. This way, primitives build on lower ones and add functionality. A primitive can also use several others, such as the trickle primitive which uses both broadcast and unicast for data dissemination. An example of the Rime stack in use is seen in Figure 2.12.

Rime uses Rime channels represented by an 16-bit integer and must be the same in both sender and receiver. It can be thought of as an identifier for Rime so that it knows how to decode the packet header to the corresponding primitive. A connection is opened, specifying parameters such as the channel and pointers to callbacks. Callbacks are invoked by the primitive, e.g. when receiving a packet.

uIP The uIP (micro-IP) protocol stack [23] offers TCP/IP-support on very small embedded devices. It comes with support for header compression and is certified for IPv6. The layered structure of the communication stacks in Contiki allows for IP packets to be transmitted over Rime and vice versa.

Examples of Primitives in Rime

Anonymous Broadcast Abc is the most simple primitive. It contains neither sender or receiver address.

Broadcast A broadcast adds the sender address packet buffer attribute on top of a abc transmission.

Unicast An unicast is a broadcast with a receiver address attribute.

Flooding Flooding uses broadcasts with the polite gossip mechanism [42] to spread a message across a network. A node initiates the flooding with a message and a unique sequence number generated by the primitive. The primitive checks if this is the first time it receives a packet with this sequence number, and if so retransmits it, otherwise suppressing it.

Route Discovery The route discovery primitive is used for finding a multihop route from one node to another. The requester floods the network with a request for a route to the target, for each hop adding the addresses of the nodes the request has passed by, and the target answers with an unicast following the chain of addresses back to the requester. It opens two channels: one flooding for the route request, and one unicast for the route ACKs from the target back to the requester..

COOJA Network Simulator

COOJA [46] is a simulator developed at SICS for simulating wireless sensor nodes. It can simulate Contiki code on various platforms and uses MSPSim for simulating nodes with the MSP430 family microcontrollers.

Other Operating Systems for Small Embedded Systems

There are a number of OS' developed by academia and industry. TinyOS and FreeRTOS are free and open source.

TinyOS TinyOS [41] is a free and open source operating system targeted at wireless embedded sensor networks with a first public release in 2000. It is written in an extension to C called nesC. TinyOS is used in academic research and industrial applications. The kernel is event-driven and being completely non-blocking (except for Tasks), it uses callbacks for every operation that lasts

longer time than some hundred microseconds. It has been ported to about a dozen platforms.

FreeRTOS FreeRTOS [54] is a real time OS released under a GPL-licence with the option of having closed proprietary source code. It is coded mostly in C, with some architecture specific parts in assembler. Designed to be easy and small, it is ported to many platforms such as Atmel AVR, ARM7, ARM9, MSP430, some PICs, x86, 8052 and more. It is also kept in a version with identical code base called OpenRTOS, the only difference being the licensing. It supports preemptive, prioritized multithreading; TCP/IP support can be added with uIP from Contiki.

Chapter 3

Politecast

Politecast is a new communication primitive that assumes that nodes that are interested are listening. By transmitting only to nodes that are interested, it offers less congestion, less overhearing, lower latency and a shift of the transmission burden. It is an addition to the design space for the application designer. The name is chosen to reflect on the nature as non-obtrusive, non-intrusive and polite.

3.1 Four Problems with Broadcast

In this work I identified four problems with broadcast. They are: overhearing costs, increased congestion, biased effort and high latency.

Overhearing Costs Overhearing occurs when nodes overhear transmissions not explicitly addressed to itself. This is a pointlessly increased cost if the node is not interested in the contents of the transmission. Thus it is hindered from preserving energy by sleeping.

As the radio transceiver is the component with the highest power consumption it is common to shut it off as much as possible. In a neighbor agnostic network, i.e. one that uses broadcast frequently, the amount of overheard transmissions can have a large impact on power consumption. In parts of the Great Duck Island deployment [52] more than 25 % of the radio power consumption was attributed to overhearing. Ideally, a node that is disinterested should not be bothered at all.

Increased Congestion As wireless networks share the medium, congestion can cripple networks performance. As network size scales up, the more traffic there is and the more likely congestion and interference is to happen. Broadcast transmits repeatedly (strobes or the data packet), thereby causing much more traffic than the actual useful data packet. Collisions makes it harder to get a

message through as packets are corrupted or dropped. Congestion can be lowered by reducing transmission periodicity, but then also application performance is reduced.

Biased Effort Broadcasting in LPL and LPP is a biased, or unbalanced, effort where the sender has a higher power cost for the transmission than the receiver. In LPL, the sender will always need to transmit for the entire sleeping period, but receivers will on average only be awake for half that period. LPP is even more unjust: the sender is awake an entire period but receivers will receive the broadcast packet directly after they have probed the medium, and can then go back to sleep. From a global network perspective, this is good if few nodes are expected to broadcast as the highest power consumption for broadcast transmissions occurs in few nodes (the senders). However, if the majority of nodes transmit broadcasts, or if the node that needs to broadcast also has the largest need to save energy, it can be better if the burden is shifted to the receivers instead.

High Latency In applications like alarm systems, networks with a high need of synchronization or interaction rich systems, high latency reduces application performance. From a transmission wide perspective, broadcast is slow as it is limited by the sleeping schedules in receivers.

3.2 Politecast

Politecast assumes interested nodes are listening 3.1. The data packet is sent just once, without wake up strobes. Other nodes are not awakened. The policy for when and for how long a node listens for politecast transmissions is controlled by the application, not the primitive. This is necessary as applications are so qualitatively different that one policy cannot be optimal for all cases. One application can have a sensor based trigger, and listens until a specific transmission is heard, while another has a periodic trigger and listens for all transmissions during a fixed amount of time.

Even though it is possible to improve broadcast in a number of ways, most of the previously stated problems remain. By numbering the strobes, a waking node can go back to sleep until the data transmission, thus lowering the power consumption with the receiver but not the sender. By tagging messages with a class identifier — e.g. sensor data, commands, parameters, neighbor discovery — this tag could be included in the strobe, lessening the overhearing when disinterested problem but not others. A node waking up could read out from the strobe that there is a soon to come sensor data message, and if it is not interested, it can go back to sleep. Instead of transmitting strobes, the sender can repeatedly transmit the actual message. A node waking up can receive the packet and go to sleep until the sender is done transmitting, but the bulk part of the effort still lies with the sender.

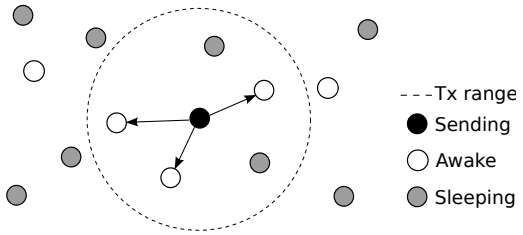


Figure 3.1: Politecast assumes that if a node is interested, it will be listening for the transmission. Hence, no waking up strobe transmissions or waiting for node probing are performed.

Politecast takes a different approach: instead of pushing the message to the receivers, it transmits the data packet immediately without any attempt to wake up nodes. It assumes that any interested receivers are awake and listening. No waiting is performed and no repeated transmissions. As the transmission is immediate, the latency is as low as possible to achieve. A receiver must explicitly turn the radio on to listen. Therefore, nodes will only receive politecasts if they are listening, either by politecast or by the underlying power saving MAC protocol. It scales well even with dense networks because of the small amount of traffic. As the sender does not make any effort in waking up or waiting for nodes, the effort is as small as possible for the sender.

3.2.1 Example

When in a network with mobile nodes, it is often desirable to know who your neighbors are. This can be used for sharing resources (e.g. node B sends temperature data to node A) or finding routes to a distant node with multihop transmissions. Neighbor discovery is a costly service as every node must announce itself to the others by sending periodic "Hello"-messages. It becomes a tradeoff between lifetime and performance. By sending and listening more often, the lifetime is shorter but the accuracy is higher and latency is lower, as long as there is no or low congestion. Also, a node might not always be interested in knowing the neighbors all the time, but only when it is moving. It should therefore be able to adjust to this change of situation.

A future scenario: a person is wearing a number of devices with wireless communication. Every node transmits beacons periodically with an identifier and information about what services it can offer. When the person adds a new device, e.g. mounts a heart pulse sensor, it upon bootup listens for new devices. It finds the other devices and share heart rate data with them. When the person start moving, one of the devices detects the movement with an accelerometer and starts listening for new neighbors along the way.

Chapter 4

Implementation

The politecast implementation covers two layers: the network layer and the MAC layer. At the network layer, politecast needs timers and data structures for the connection. At the MAC layer, politecast needs to be able to switch on the radio for listening.

This politecast implementation in Contiki OS is very lightweight. It adds little overhead in terms of computing and memory usage. The code size and static RAM usage is low: 328 bytes ROM and 32 bytes RAM is needed for one politecast connection and every extra politecast connection adds 32 more bytes RAM.

4.1 Network Layer

Politecast builds on top of the broadcast primitive. When transmitting, politecast adds a politecast packet buffer attribute to a broadcast packet and invokes the broadcast transmit function. There is a connection-local transmission power setting (TxP) so that politecast sets the transmission power before transmitting, instead of requiring the user to do that for every transmission.

When a politecast packet is received, the underlying primitives will invoke the registered receive callbacks which ends up in the callback registered in the application when the connection was opened. When a politecast packet is received, the packet is passed on through the Rime stack, each primitive stripping the packet of its attributes respectively. Finally, the politecast receive callback function is invoked, in which the packet can be read.

4.2 MAC layer

Politecast requests the MAC protocol to switch on the radio when listening. Listening is either for a limited set of time, or for a non-specific duration. When a time is specified, a timer is started that when expired tells the MAC protocol that it is finished listening and sleeping can resume. If the listening is undefined,

listening is started without the timer and listening would only be stopped by a call to the listening-stop function. The radio must be kept on for the entire listening period. When listening is turned off, the MAC protocol will control radio duty cycling. Rime offers functionality to switch on or off the MAC layer duty cycling. When transmitting and the politecast attribute is set, the packet is sent once and immediately without any wake up strobes.

The listings 4.1 to 4.5 show pseudo-code for the most important functions of politecast. The C code API is presented in appendix A and code examples in appendix B.

```
Politecast send()
    set Rime politecast attribute
    set transmission power setting
    invoke lower layer transmit mechanism
```

Listing 4.1: Transmitting with politecast

```
Politecast receive packet from underlying layer/primitive
    if such exists:
        invoke 'receive'-callback
```

Listing 4.2: Politecast receiving packet

```
Politecast listen(time t)
    if t ≠ infinite:
        start timer with expiration time t
            on expiration, invoke politecast listen stop()
    request MAC protocol to switch radion on
```

Listing 4.3: Politecast listening

```
Politecast listen stop()
    check flags
        if we are listening:
            if timer is running:
                stop expiration timer
            request MAC protocol to regain control
```

Listing 4.4: Politecast stop of listening

```
Politecast listening timer expired
    give back control to MAC layer
    if such exists:
        invoke 'timeout'-callback
```

Listing 4.5: Politecast listening period expired

Chapter 5

Evaluation

Politecast was evaluated with experiments and simulations in three case studies. As politecast is qualitatively different from broadcast, there were inevitable differences between implementations.

In the power saving MAC protocols used in the evaluations, there is a trade-off between lifetime and performance, affecting latency and congestion. By tweaking the MAC protocol timings, the sleep periods can be shorter or longer, resulting in less or more repeated transmissions for a broadcast. For politecast, transmitting more often means shorter listening periods can be used.

5.1 Method

The three case studies were simulations conducted in a network simulator (COOJA), in-field experiments with real hardware and deployment-like conditions and microbenchmarks conducted both experimentally on real hardware as well as in simulations. The three case studies are Steal the Light, Neighbor Discovery and the Lega System. They represent different kinds of application domains with different characteristics and metrics.

5.2 Metrics

The four problems with broadcast can for the most part not be measured directly, and measurable metrics must be used as proxies.

5.2.1 Overhearing Costs

Overhearing when being disinterested is a waste of power which is measured by radio utilization and amount of overheard traffic (number of received packets). The radio is generally the highest power consuming component in motes, and all other components power consumptions are small in comparison. The

power consumption differs in listen and transmission mode, therefore both are measured.

5.2.2 Increased Congestion

With more nodes in a denser network follows more radio traffic, increasing the risk of congestion. This is measured by change in radio utilization. Congestion will cause an increase in listen mode utilization as a node will await a free medium, and a decrease in radio transmission utilization as the radio CSMA-CA mechanism backs off and waits for CCA. If CSMA-CA do not get CCA within a defined period of time, the transmission will be dropped.

5.2.3 Biased Effort

The biased effort effect on power consumption is measured with radio utilization. Both transmission and listen utilization are measured.

5.2.4 High Latency

Latency is measured as the time between when an event at the earliest could occur, and when it really did (as in ideal versus real neighbor discovery latency), or the time between two following events (start to finish of a transmission).

5.2.5 Other

Other metrics measure qualities that are hard to capture with the above factors. The purpose with this is to shed light on the qualitative differences between politecast and broadcast. E.g. accuracy for the neighbor discovery case study can indicate how suitable politecast is compared with broadcast if a reliable service is required.

5.3 Simulation Setup

Simulations were used where ground truth measurements were needed but too hard to acquire with experiments. For example, for neighbor discovery, the latency and accuracy are important and in order to measure them the position is necessary, which is hard to get with high enough spatial resolution with many nodes moving fast at the same time.

5.3.1 COOJA Simulation Setup

The simulated radio medium was the unit disc graph model (UDGM), which means that three things can happen in collisionless communication: successful transmission, probabilistically successful reception proportional to distance and transmission power, or unsuccessful transmission (Figure 5.1). UDGM is a simple model (see Section 2.1.3) but it fulfills the purpose of the simulation as

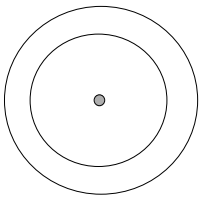


Figure 5.1: The node has 100 % transmission success within the first radius and receives interference within the outer radius, resulting in a lower probability for a successful transmission.

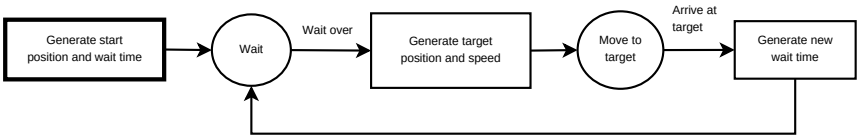


Figure 5.2: The principle behind the random waypoint mobility model.

it makes comparisons between implementations easier when no environmental effects are taken into account.

5.3.2 Position data, RWMMSim

The COOJA simulator accepts position data for the simulated nodes. I wrote a WSN mobility simulator that can generate such position data for any number of nodes moving in 2D-space: RWMMSim. RWMMSim uses the random waypoint mobility model (RWMM) [22] and is highly parameterizable, e.g. speeds, time durations, space dimensions. It also outputs statistics and other data, e.g. the time when nodes are in or out of range and what distance each node has travelled. The beginning of the simulation is discarded to avoid a large number of simultaneously stationary nodes.

RWMMSim initially generates random starting positions and wait durations. When the wait is over, a speed and target position is generated. When target is reached, the node waits for a random time before starting over with a new target (Figure 5.2). Figure 5.3 shows an aggregate of all the positions the simulated nodes appear at during the duration of the simulation.

I chose parameters that are plausible for people moving in a large confined space, with speeds approximately between slow walking and fast running. The full set of parameters is shown in Table 5.1. The same position data was used in all neighbor discovery simulations.

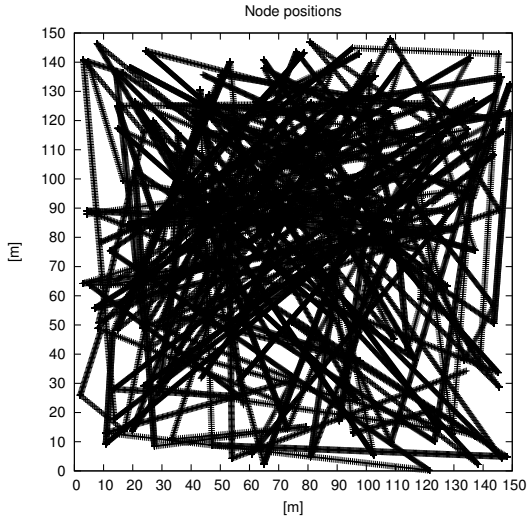


Figure 5.3: All the generated positions of the nodes in the simulation.

Time [s]	600
Minimum speed [m/s]	1.0
Maximum speed [m/s]	4.0
Minimum wait time [s]	2.0
Maximum wait time [s]	10.0
Time step [s]	0.2
Number of nodes	15
Size [m^2]	$150 * 150$

Table 5.1: Parameters for generating position data in RWMMSim.

5.4 Case Study: Steal the Light

Steal the light is a token passing-like application with low latency requirements.

5.4.1 Steal the Light Principle

In Steal the Light, one node has the light (henceforth the "keeper"), and other nodes that do not (henceforth called "stealers") will try to come close to steal the light. As this is an application meant to be used in a playful setting, it is important that the delay between coming in range to stealing the light is low. A long delay will make the feature seem slow and unresponsive. Steal the Light can be said to be the inverse of the playground game "tag" and is illustrated in Figure 5.4.

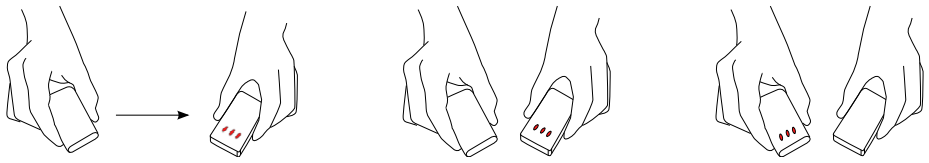


Figure 5.4: Steal the light: (*left*) node A (to the right) has the light while B (to the left) moves closer. Periodically, A transmits beacons with a low transmission power setting, thus having a very short range, telling other nodes that it has the light. (*middle*) B is in range and receives the beacons from A. It sends a request to steal the light. (*right*) A surrenders the light to B.

Parameter	Broadcast	Politecast
MAC protocol	ContikiMAC	ContikiMAC
Channel check rate [Hz]	16	16
TxP [dBm]	-25	-25
Beacon period [s]	0.5	0.5
Light keep time [s]	1	1

Table 5.2: Settings for Steal the Light, broadcast and politecast versions.

5.4.2 Implementation

The light is kept for a minimum amount of time, and is only passed on if the nodes are stationary. Both keeper and stealer transmit unaddressed beacons to spread node state information to neighbors in range in order to reduce latency and increase accuracy (not missing an encounter). A stealer hearing a keeper requests the light, and a keeper hearing a stealer will offer the light, which will then be followed by a request. The request is answered by a go-ahead, concluded by an acknowledgment. The beacons are transmitted with broadcast or politecast respectively, but all other communications are unicasts. The settings are shown in Table 5.2.

Politecast The only differences between the implementations are that in the politecast implementation beacons are transmitted with politecast and a listening period is used. The listening is triggered by sensor readings. When the accelerometer senses that the user stopped moving, it triggers a listening for a fixed amount of time. The listening period is slightly longer than the beacon period, 505 ms. After having given up the light, the node starts a short listen just before the minimum keep time is over. That way it will hear the first beacon from the keeper, should they still be in range.

5.4.3 Metrics

Power Consumption Radio utilization is here a proxy for power consumption. The total (Rx and Tx) radio utilization is measured with energest. Both listen and transmit utilization are measured.

Congestion Congestion is measured by the radio utilization in transmit mode (Tx mode). Per node, the total Tx mode utilization is divided with the number of token passings, and the mean taken over both nodes is normalized with politecast as base. The radio utilization is measured with energest.

Latency Delivery latency is the time from transmitting a beacon until having delivered the light. It is measured locally on each node with software timers having a $\frac{1}{16384}$ s resolution.

5.4.4 Simulation Setup

Two simulations were conducted. In the mobile scenario, two nodes were used, of which one started with the light. They moved in a predefined pattern with varying stopping times between 10–60 seconds, using the Mobility plugin in COOJA. The same movement data was used for both broadcast and politecast. As the accelerometer is not simulated in COOJA, the listening was instead triggered by a listening scheduler synchronized to the movement data from Mobility to trigger when the node stops.

In the static scenario, two nodes were situated close to each other and constantly swapping the light between them. The rationale for this simulation was to see how the tradeoff between lifetime and performance would compare between implementations when the beacon rate and the movement intervals varied. The movement intervals were 5, 10, 20 and 40 seconds with fixed beacon rate of 2 Hz. The beacon rates were 0.5, 1, 2, 4 Hz with a fixed movement interval of 10 seconds. The lifetime is calculated with a Tmote Sky mote as basis, using two 2200 mAh batteries and neglecting the power consumption for all other peripherals.

5.4.5 Results

Using politecast, lifetime is longer and latency, congestion and overhearing lower compared with broadcast.

Power Consumption Politecast lowers the power consumption with up to 74 % compared with broadcast (Figure 5.5). Politecast gets a longer lifetime when the beacon rate is lower, while the opposite is true for broadcast. The reason for this is that with a higher beacon rate, the listen time can be shorter while broadcast has a fixed cost. When the movement interval increases, the politecast implementation gets a longer lifetime as there is less listening triggered. Broadcast has approximately the same for all movement intervals. This

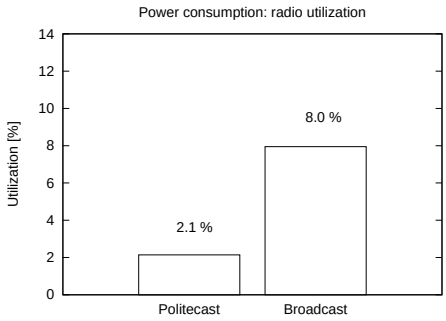


Figure 5.5: Broadcast has a much higher power consumption than politecast due to the increased amount of transmissions forcing the other node to listen more (Figure 5.8). The power consumption is measured as total radio utilization over time.

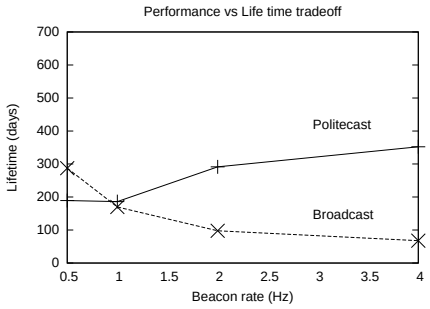


Figure 5.6: Lifetime depending on beacon rate. As politecast need to listen for a longer time with decreasing beacon rate, it also gets a shorter lifetime. Broadcast gets a shorter lifetime when beacon rate increases as it transmits more data.

shows the importance of having a good strategy for when to listen and for how long.

Congestion Politecast lowers radio transmission utilization for every token passing with 95 %, compared with broadcast (Figure 5.8). This shows that broadcast transmits much more than politecast, and thus politecast is more suited for dense networks as it can lower congestion.

Latency The politecast implementation is more than twice as fast as the broadcast (Figure 5.9). The mean delivery latency for politecast is 287 ms and for broadcast 606 ms. The poor performance with broadcast can shatter the experience in an interaction rich application.

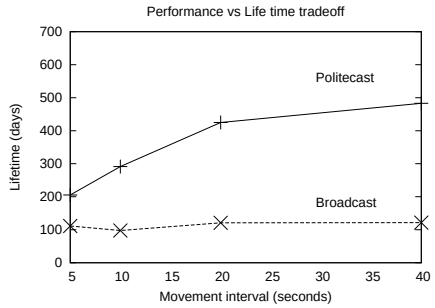


Figure 5.7: Lifetime depending on mobility. The more often the simulated node moves, the more often politecast has to listen, thus shortening lifetime. Broadcast has the same lifetime for all intervals as it has no movement based trigger.

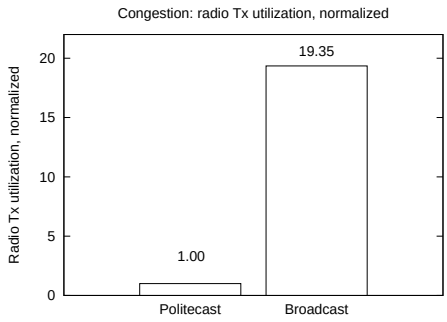


Figure 5.8: Politecast has a much lower congestion than broadcast. Broadcast transmits up to more than 19 times as much as politecast for every token passing.

5.5 Case Study: Neighbor Discovery

Neighbor discovery is an important application domain with great importance, especially as more and more devices are mobile.

5.5.1 The Neighbor Discovery Primitive

Contiki has a neighbor discovery primitive (henceforth called ndBROAD) that uses periodic broadcasts to advertise itself to other nodes.

Implementation with Broadcast ndBROAD uses broadcast, so there was no modification necessary but choosing the parameters that would control timings. They are parameters for the primitive, and parameters for X-MAC. The primitives parameters are the min_period and max_period. After a first period of *init_period* the advert is periodically transmitted at a random time between

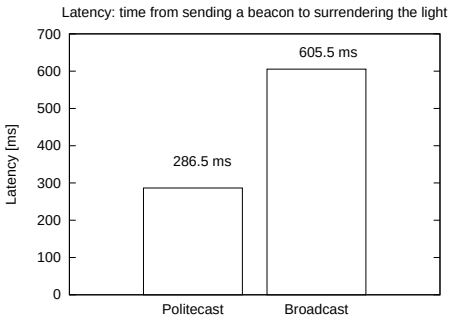


Figure 5.9: Politecast is more than twice as fast as broadcast. The latency is measured from transmitting the beacon until the light is passed on.

Parameter	Sim1	Sim2	Sim3	Sim4	Sim5	Sim6
On time [ms]	5	5	10	5	5	5
Off time [ms]	58	58	115	495	58	495
Strobe time [ms]	80	80	135	595	80	595
Strobe wait time [ms]	4.4	4.4	8.8	4.4	4.4	4.4
init_period [s]	0.5	1	0.5	0.5	2	2
min_period [s]	0.5	1	0.5	0.5	2	2
max_period [s]	5	3	5	5	7	7

Table 5.3: X-MAC settings for ndBROAD in simulations.

min_period and *max_period* to avoid collisions. The parameters for X-MAC (see Figure 2.7) control sleeping and transmission timings. Lowering them makes broadcasts faster, as fewer strobes are transmitted. It follows that the medium must be checked more often so that no transmission is missed. The parameters for the simulations are shown in Table 5.3.

Implementation with Politecast The implementation with politecast (henceforth called ndPOLITE) transmits periodic beacons, with politecast. Listening was triggered by a periodic timer. For ndPOLITE, there are three parameters beyond the X-MAC settings — *advert_period*, *listen_length* and *listen_period*. They set how often adverts shall be sent, how often to trigger listen and for how long to listen. As in ndBROAD, beacons are transmitted at a random time within the listen_period in order to minimize the risk of successive collisions.

5.5.2 Metrics

Congestion Congestion is measured with the number of radio collisions.

Parameter	Sim1	Sim2	Sim3	Sim4	Sim5	Sim6
On time [ms]	5	5	5	5	10	5
Off time [ms]	58	495	495	495	495	495
Strobe time [ms]	80	595	595	595	595	595
Strobe wait time [ms]	4.4	4.4	4.4	4.4	8.8	4.4
listen_period [s]	5	5	10	1	10	10
listen_length [s]	0.28	0.28	0.75	0.125	0.28	0.28
advert_period [s]	0.25	0.25	0.69	0.5	0.25	0.25

Table 5.4: Settings for ndPOLITE in simulations.

Power Consumption Power consumption is measured by estimating the radio utilization with energest and COOJA. By measuring utilization rather than power, the result can be directly applied on other hardware with differing power consumption.

Latency Latency is measured as first and full discovery latency by cross referencing the position data from RWMMSim with the timestamp of when a node receives a beacon. First discovery latency is the time between nodes coming in range and one node discovers the other. Full discovery latency is when also the second node discovers the first.

Accuracy Accuracy is a metric of how well the implemented neighbor discovery works at discovering neighbors. The more missed discoveries, the lower the accuracy. Partial discovery is when only one of the two nodes discover the other, full discovery is when both discovers each other. As with latency, the position data from RWMMSim used with the ranges set for UDGM in Table 5.5 is used to calculate if and when two nodes should discover each other, and the log from each node shows whether they did or not.

5.5.3 Simulation Setup

15 nodes were simulated, moving in a confined space. In order to see how an indoor environment with high versus very low RF attenuating characteristics would affect the outcome, I ran the neighbor discovery simulations twice — once with the full reception-radius set to 51 m and once set to 75 m as seen in Table 5.5. In the 75 m setting, the transmissions reach farther, meaning more interference and collisions but also potentially more discoveries.

5.5.4 Results

Power Consumption Most noticeable is how the radio utilization varied in between implementations (Figure 5.10). For broadcast, the maximums were on average 21 % higher than the minimas, for politecast it was 3.4 %. The large

Radio reception range [m]	50 and 50
Radio medium model	UDGM
Radio reception range [m]	50 and 50
Radio interference range [m]	51 and 75
Time limit [s]	600
Random seed	123456

Table 5.5: Parameters used in COOJA and RWMMSim for the Neighbor Discovery simulations.

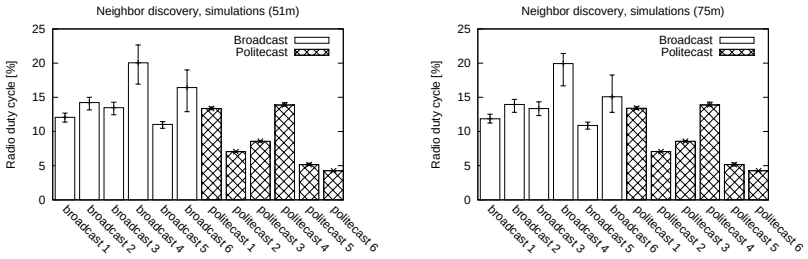


Figure 5.10: The average radio utilization in neighbor discovery. The utilization for politecast varies by a small amount from the average values, as expected as politecast does not wake up neighbors. Transmission radius (*left*) 51 m, (*right*) 75 m.

variation in utilization is because of the overhearing problem, and the small variation for politecast shows how the politecast implementation is unaffected even in dense networks. This means that a service or application can be designed with politecast to use a predefined radio utilization, which will not deviate much from the set value even if there are many neighbors or high congestion. As it will not wake up nodes, it should also not affect them much.

Congestion Politecast has fewer collisions than broadcast, except for Politecast1 which has the highest advert rate. As could be expected, the amount of collisions increase when the reception radius increases (Figure 5.11). With the reception radius increased from 51 to 75 meters, this is ca 50 % longer range, but 125 % more surface coverage. The effect is similar to increasing transmission power, resulting in an increase in congestion. Both implementations scale equally much proportionally, on the order of 10–20 times as many collisions in the 75 meter case than for 51 meter.

Latency As congestion increases, latencies also increases (Figure 5.12), for both broadcast and politecast implementations. As politecast transmissions are direct, this highlights the listen policy problem. Choosing when and for how

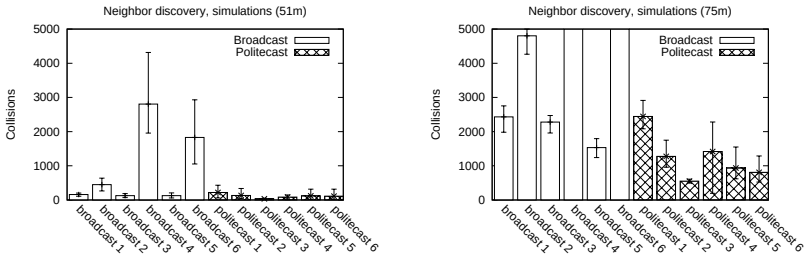


Figure 5.11: The plots shows average total amount of collisions for the two transmission radii. (*left*) 51 m, (*right*) 75 m.

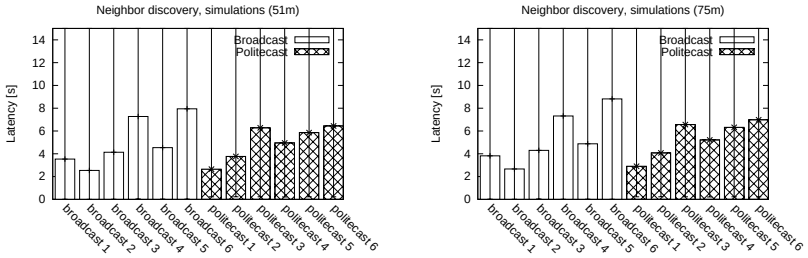


Figure 5.12: The average latency of neighbor discovery. Transmission radius (*left*) 51 m, (*right*) 75 m.

long to listen has a great effect on performance and lifetime.

Accuracy The broadcast implementation has generally higher accuracy (especially Broadcast2), because it forces nodes within listening distance to wake up, at the cost of reduced lifetime and increased congestion. Figures 5.13 and 5.14 show the accuracy of neighbor discovery at the two different radii. Broadcast gets a lower accuracy when the radius is increased, due to collisions. Politecast is not as sensitive and maintains approximately the same accuracy. However, for both primitives, the effect on accuracy from increasing range is small, on the order of 1–3%.

5.6 Case Study: The Lega System

The Lega [40] ("Lega" is a Swedish word for the temporary and visible trace in the vegetation after an animal that has been lying down) was used for exploring and enabling richer group interaction and bodily expressions in the context of an art exhibition. Every year, the art gallery Liljevalchs [8] hosts an art exhibition called Vårsalongen which takes place during two months in the spring. We, the

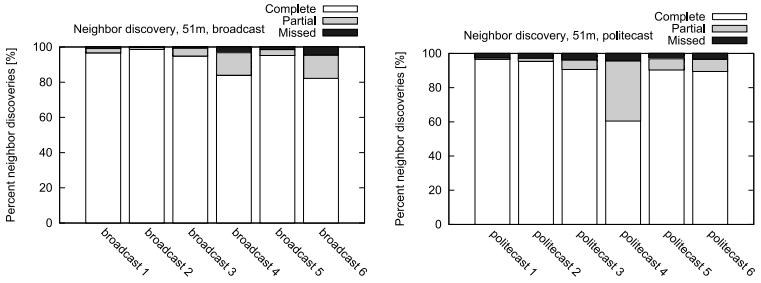


Figure 5.13: The accuracy of the neighbor discovery at 51 meter reception radius. (*left*) Broadcast, (*right*) politecast right.

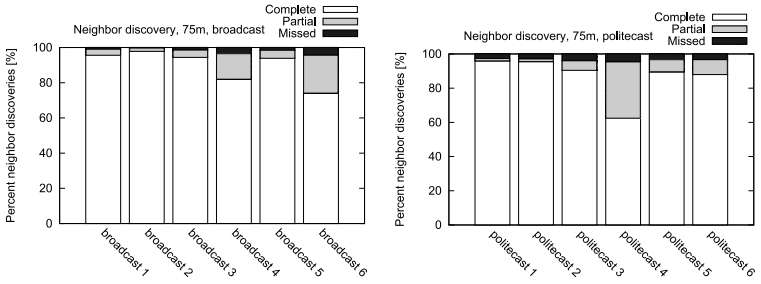


Figure 5.14: The accuracy of the neighbor discovery at 75 meter reception radius. (*left*) Broadcast, (*right*) politecast right.

Designing Supple Systems project group, ran the Lega system with users during the entire exhibition.

Every participant in a group of 2–5 persons is given a device, called Lega. The user can then interact with the group by forming, leaving and finding "traces". By interacting with the Lega the user forms the trace, a message containing a digital version of that interaction.

The Legas are built around Sentilla JCreate nodes. Vibration motors, touch sensors and LEDs are connected to the expansion port in the bottom, via port expanders. A servo motor is placed on the top side and worked as a controllable press button and indicator.

32 static infrastructure nodes (henceforth "IS nodes") are placed throughout the exhibition. They transmit radio beacons, which are used for approximating a position in the Lega. They also keep the traces and gather data for off line analysis and statistics. The trace is transmitted to the IS node that is closest, in which it is saved until another Lega enters that approximate position. Then it is transmitted to the Lega and "played back" with LEDs and vibrations. From knowing who the sender was, seeing and feeling the trace played back, the receiver makes an interpretation of what they sender might have wanted to communicate. See Figure 5.15 for an illustration of the system, and the picture

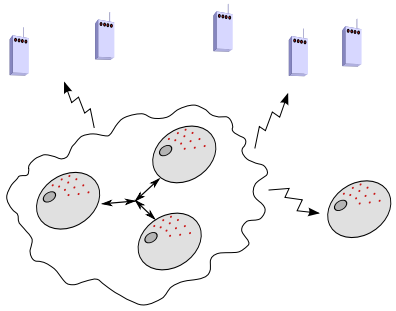


Figure 5.15: Lega system overview, showing the personal devices (Lega) and the static infrastructure nodes used for positioning and storing traces.

on the front page of the thesis for what the Lega looks like when it is leaving a trace.

Being hidden in the walls, they are hard to reach. Thus, changing batteries on the 32 IS nodes is expensive, time consuming and cumbersome, so the radio must be turned off as much as possible, while still ensuring performance. As they do not use LEDs or sensors, the radio is the highest power consumer.

Positioning

Position awareness is a fundamental corner stone of the Lega system. A Lega should not play a trace if it is not on that location where the trace was left. A spatial resolution of about half a small room is sufficient (corresponding to a few meters). By having more IS nodes with a shorter transmission range, a higher resolution can be achieved.

The basis for approximating the position is the beacons transmitted by the IS nodes every second. A Lega approximates the most probable location from the number of beacons received per time unit, which was received last and the RSSI. Every third beacon also contains information on eventual traces saved on that node.

The more received beacons, the more data for the software to approximate a position from. This is because in theory there should be a good correlation between RSSI and distance, but in practice this is only true for an average on many samples [38]. For the Lega system, I wrote a software library called Position with which the Lega keeps a neighbor table based on the received beacons. With a function call, the table is searched for the most likely position.

Difficulties with Positioning Overhearing and an unpredictable environment were the biggest concerns when getting the positioning to work reliably, with low latency and with high accuracy. The Liljevalchs art hall is built mainly out of bricks and wood, making it practically transparent to RF. Beacons must be transmitted often enough for positioning with low latency when the user

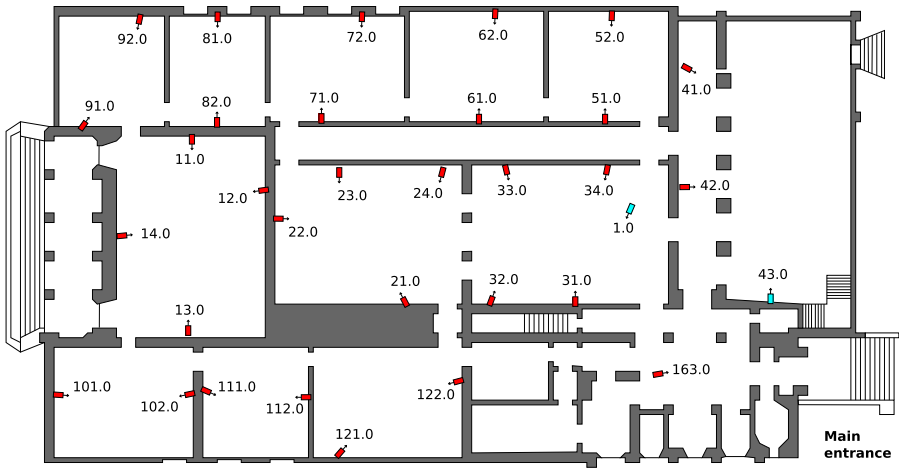


Figure 5.16: The map shows the position and orientation of the infrastructure nodes (Tmote Sky) at Liljevalchs bottom floor. The arrows point in the direction the PCB PIFA $\frac{\lambda}{4}$-dipole antenna was directed; i.e. the USB connector pointed in the opposite direction. As we noticed on site, even though the application note for the PIFA antenna [6] stated that it is fairly omnidirectional, the orientation seemed to matter much. We did not do any measurements on this though. Two nodes were on different floors: 1.0 was in the basement and 43.0 was one floor up, with view over the large and open sculpture hall.

moved around. Beacons must also be transmitted with a fairly high transmission power as many objects absorb the radio signals: the visitors, the hands of the user, and the Lega shell itself. The number of visitors varied much and fast: a $5 * 6$ m^2 room could in $\frac{1}{2}$ minute go from 25 to zero visitors and vice versa. People walked in groups and often clustered. Most IS nodes were hidden in small cupboards in the walls, ca 20 cm above floor level at positions according to Figure 5.16.

As a result, a high beacon rate together while at the same time avoiding congestion and ensuring long lifetime were design goals. Following this reasoning based on empirical observations, IS nodes was set to transmit beacons once a second with a transmission power setting of -7 dBm. This turned out to be a good compromise between having low enough transmission power to minimize overhearing between rooms, but also high enough to be received.

5.6.1 Implementation

Broadcast Version In order to achieve low latency and transmit one beacon per second, I changed the settings of the X-MAC protocol according to Table 5.6. Now, it checked the medium more often, enabling the use of a shorter strobe train at the cost of a shorter lifetime. This makes for less congestion and lower latencies. Otherwise, the implementation is exactly as the politecast version

On time [ms]	5
Off time [ms]	244.5
Strobe time [ms]	295.5
Strobe wait time [ms]	4.4

Table 5.6: X-MAC settings for broadcast version.

On time [ms]	5
Off time [ms]	495
Strobe time [ms]	595
Strobe wait time [ms]	4.4

Table 5.7: X-MAC settings for politecast version.

except that a broadcast connection is used. Transmitting a beacon would take ca 0.3 seconds, leaving 0.7 seconds until the next beacon.

Politecast Version The politecast version transmits a beacon every second using the X-MAC settings in Table 5.7. This would make for a good tradeoff, as the number of transmissions that would use strobes (unicast and broadcasts) was anticipated to be rare in comparison.

The listen policy could have been based on a motion trigger (accelerometer), but instead the Legas are always listening. As the Legas are high power consumers, the radio power consumption is only a small fraction of the total. When running idle, sensors are constantly being read and some LEDs are on making the power consumption reach about 0.5 W. When playing a trace the peak power consumption reaches 3–6 W for hundreds of milliseconds as the servo and vibration motors starts and more LEDs are lit up.

5.6.2 Metrics

Overhearing Costs Total radio utilization is the primary metric for power consumption for the IS nodes and measured with energest.

Congestion Congestion is measured with radio Tx utilization and the number of overheard transmissions. By comparing the radio Tx utilization with a known base case – a node with no neighbors – the effects of congestion can be measured. A lower utilization means that transmissions has been dropped or backed off, delaying other transmissions thus reducing the Tx utilization. The number of overheard beacons should ideally be zero as no IS node is interested in any other IS node.

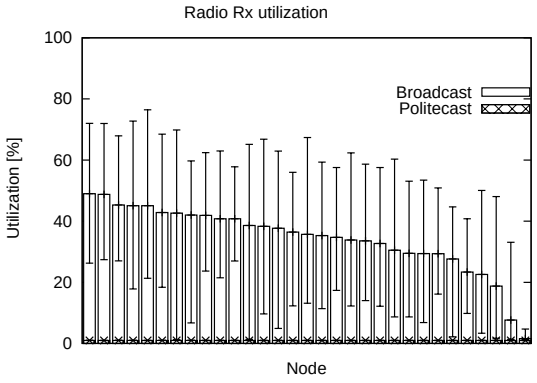


Figure 5.17: The average radio Rx utilization, which is the most power consuming mode (for CC2420, nearly 10 % higher than transmitting at full power).

5.6.3 Experimental Setup

For each implementation the infrastructure was running for one hour using each implementation, on location without any visitors or personnel in the exhibition halls. Empty halls was the most common case during the exhibition opening hours as most visitors came in the evenings and it was mostly empty during the day. The IS nodes were placed in the same positions as during live user studies, as seen on the map in Figure 5.16. All nodes except node 41.0 were operational.

5.6.4 Results

Overhearing The politecast implementation has much lower overhearing costs than broadcast. The cost is also consistent (figures 5.17 to 5.21). Most broadcast IS nodes have 25—35 times higher Rx utilization than politecast. As politecast does not wake up neighbors, the average utilization is consistent no matter how many neighbors a node has. The makes it possible to predict lifetime. For broadcast, the nodes with the fewest neighbors also had the lowest radio utilization, clearly showing the problem with overhearing. The broadcast node with the lowest Rx utilization (Figure 5.17) was placed in the basement and had no neighbors.

Both politecast and broadcast versions overhear beacons, but politecast much less than broadcast (Figure 5.19). The reason for this is that broadcast wakes up the neighbors no matter if they are interested or not. Politecast does not wake up neighbors and has far fewer overheard beacons than broadcast. It is not zero as nodes are using X-MAC and has a 1 % listening duty cycle, thus any overheard beacons occurred during such a wake up period.

Congestion Politecast has a low and consistent radio Tx utilization, meaning that congestion is low and predictable. This is in contrast to broadcast where

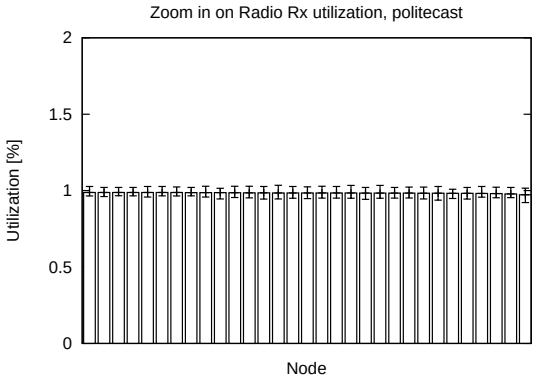


Figure 5.18: The radio Rx utilization for politecast is consistent and low, no matter how many neighbors a node has.

Tx utilization is on average about 30 times higher and varies depending on the number of neighbors of the node (Figure 5.20). Ideally, all nodes in each implementation respectively should have the same Tx utilization as they transmit beacons with the same packet size and periodicity. Clearly, broadcast had problems. The node with no neighbors transmitted the most, approximately five times more than the least successful node, which had many neighbors. The cause of this behavior is congestion, as the CCA detects energy and CSMA-CA backs off the transmission. In contrast, all the nodes in the politecast version have a low and consistent radio Tx utilization (Figure 5.21). It is low because it does not transmit strobes and consistent because congestion is very low, not triggering CSMA-CA backoffs.

Robustness The broadcast implementation was found to have several outages of service during the evaluation period, whereof one node was down more than 45 seconds. Politecast had none such outage period. The positioning service needs to hear beacons in order to approximate a position. Long durations of suppressing transmissions due to congestion can cause severe application performance degradation. If CCA detects energy, CSMA-CA will first suppress the transmission. If energy is repeatedly detected when trying to retransmit, it will drop the packet, meaning the transmission will never be sent.

If beacons are suppressed or dropped, the Lega will not be able to save traces at the right IS node or find other users traces when passing by, and the user experience will be worse. Here, a 15 second time period was considered critical during which beacons had to be heard if the user was at that location, or else the service was accounted for as being down. For politecast, the service never went down on any node. For broadcast, the worst coherent down period was found to last for more than 45 seconds and the worst node had in total 18 such periods during the measurement period of one hour, meaning a 7 % down time

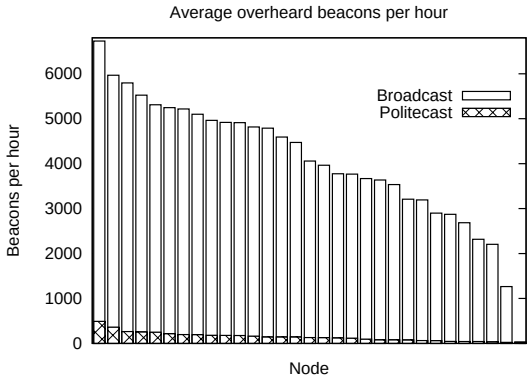


Figure 5.19: The number of overhead beacons per hour on average. This shows how broadcast wakes up neighbors, even if they are not interested in the information being transmitted. The typical broadcast node has 10—25 times the number of overheard beacons than politecast.

(Figure 5.22).

By comparing the amount of transmissions to the ideal amount, broadcast was found to have suppressed transmissions in all nodes but one (Figure 5.23). The worst performing node dropped almost 80 % of the transmissions. For politecast, all nodes transmitted all beacons and had no packet drop.

In the broadcast version, the effects of extreme congestion are higher total radio utilization: the listening utilization increase as the backoff mechanism waits for a silent medium, and the transmission utilization decrease as the node is suppressing the transmissions. The net sum is an increase as the decrease is small in comparison to the increase. Figure 5.24 shows the measurements from a small two-node experiment. The experiment settings are extreme, but more clearly shows the effects of severe congestion and confirms the measurements from the IS nodes.

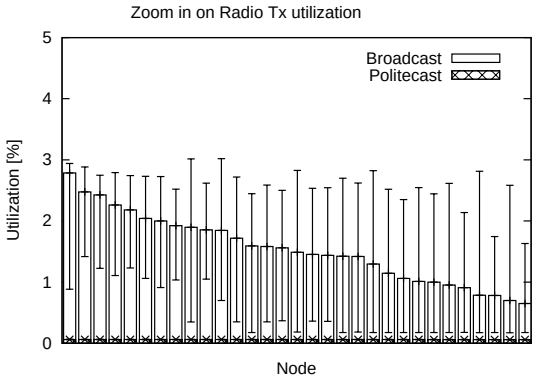


Figure 5.20: Utilization of radio in Tx mode. The ideal utilization for broadcast is around 2.8–3 %. Less than that means that congestion forced the node to back off or suppress transmissions.

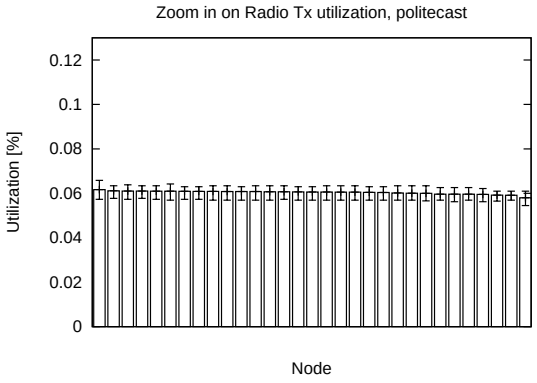


Figure 5.21: The politecast version is low and consistent on radio utilization in transmission mode. This indicates that congestion was very low as transmissions rarely or never was forced to back off by CSMA-CA.

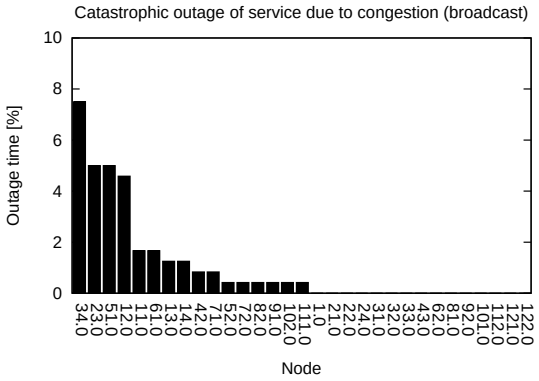


Figure 5.22: The broadcast version had catastrophic outages of service in parts of the network due to congestion. Node 34.0 was a particularly bad case with ca 8 % downtime. The politecast version had zero outage. Time resolution was 15 seconds as this was considered a critical interval.

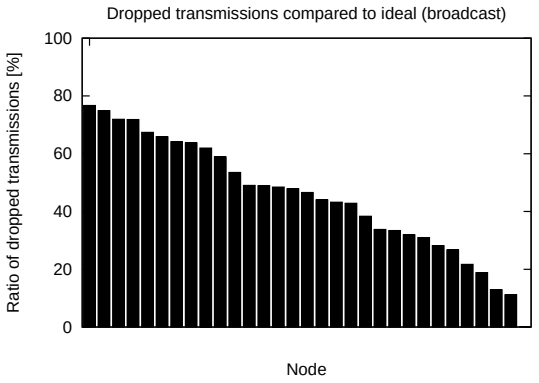


Figure 5.23: The broadcast version had problems with congestion leading to service performance degradation as beacons were not sent at the intended interval of once per second. In the plot, the ideal value is 0 % meaning no dropped transmissions.

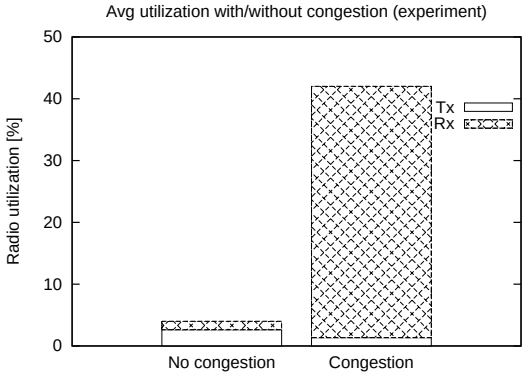


Figure 5.24: The effects of congestion on average Rx- and Tx utilization. The application is the same as in the experiment at Liljevalchs. "No congestion" refers to a single IS node and "Congestion" refers to when another node at a 10 cm distance broadcasts every 0.38 seconds, and as every beacon takes 0.30 seconds, this does not leave much room for additional traffic.

Chapter 6

Related Work

Politecast has to the best of my knowledge no peers in the body of work in related fields. Instead, there are many problem areas where politecast can be applied.

6.1 Systems

Supple Systems and LR-WPAN Low rate wireless personal area networks (LR-WPAN) are networks with devices intended to be worn on a person. Figure 2.1 shows an example of a person with several devices in the clothes, phone etc. They are likely to be highly physically and logically mobile, have unpredictable movement patterns and connectivity [29]. Networks are heterogeneous, non-uniform and rapidly changing. This means that any transmissions should be as fast as possible and opportunistic. Also, neighbor discovery should be fast, efficient and an ongoing effort. Thus, anticipated lifetime is shorter than static WSNs doing sense-and-send or batch-and-send, such as the Volcano monitoring WSN described in Section 2.1.1.

As supple systems [34], feedback is desired and expected, in the form of light, sound, motion etcetera. Also, there is high probability that other networks are co-located in the same physical area, using the same radio hardware and possibly resulting in semantic interference [32].

FriendSense FriendSense [50] is a way for co-workers to express feelings and wishes in the context of the office and with other co-workers. Every user has a personal node (Sentilla JCreate), which keeps a neighbor table and periodically measures the acceleration to approximate the degree of movement. This is periodically sent to the data sink node, which relays the data over a serial connection to a server PC. The server application translates movements into parameters shaping individual avatars on the public screen: color, position and movement. The radio communication is one-way as nothing is sent back to the personal node and there is no explicit communication between the personal

nodes. Instead, the personal nodes exploit the overhearing of transmissions to the sink, but to keep congestion low the update frequency (i.e. transmissions) is low, resulting in high latency.

eMoto eMoto [51] is a mobile emotional messaging system for exploring the affective loop — when the intellectual experiences are impossible to separate from the sensual. One goal with the interaction was to be engaging physically, intellectually and socially and enable embodied interaction. The system uses a custom built input device, similar to a Stylus, to a mobile phone to enable emotion related gestures as input. The stylus has an accelerometer and pressure sensors and connects to the phone with Bluetooth. The user studies singled out some of the disadvantages with Bluetooth as it has a high latency — several seconds — when first connecting, high power consumption and loses connection forcing the user to wait for additional seconds as the device reconnects.

6.2 Positioning and Localization

MoteTrack MoteTrack [43] is an approach to estimate the position of mobile nodes in a confined space. Static infrastructure nodes are deployed at known positions. They transmit beacons which are received with a mobile node. By comparing the RSSI of the received beacons with a pre-measured signature database, the position could be approximated with an accuracy of about 1 m at best. Their work focused on making the system decentralized and robust to node failure as one main usage of the system is rescue operations in e.g. office buildings.

Doppler Shift Positioning The authors of [39] used the characteristics of radio wave propagation to track mobile nodes. Mobile nodes transmit beacons which are received at static infrastructure nodes. The static nodes measure the RF Doppler shift of the broadcasted beacon and approximate the position and velocity of the mobile node. Accuracy was reported to be as good as 1.3 m for positioning and 0.1 m/s for speed. Focus was on developing filtering algorithms that could accurately handle both fast and slow changes of the mobile node trajectories.

RSSI-fluctuation Based Positioning In [59], the authors achieved motion tracking through walls using IEEE 802.15.4 nodes scattered around a building. Because RSSI depends on the sum of all multipath components, objects moving within the building would affect the RSSI as they moved. The static nodes around the building take turns to broadcast a beacon according to a simple token passing protocol. A node overhears a beacon from another node and decides if it is next in turn to transmit a beacon. A snooping node overhears all the broadcasted beacons and measures the RSSI, which are fed to a computer that runs the object position approximation algorithm.

6.3 Programming Abstractions

Logical Neighborhoods Many programming abstractions for neighborhoods deal with homogenic networks. Logical neighborhoods [44] shifts focus to decentralized networks where nodes are heterogenic, i.e. are different in terms of hardware and software. By replacing the notion of a physical neighborhood with a logical one, a programmer can address neighbors that fulfills certain criteria, e.g. broadcast a request for temperature readings close to nodes with humidity sensors. The authors state that broadcasts are used in order to keep it simple, but does not go into detail as to any specifics on how they are actually carried out.

Hood Hood [57] is a programming abstraction for neighborhoods that uses local definitions (i.e. on each node) of neighborhoods and shared variables, called attributes. Nodes regularly broadcast their shared variables (e.g. sensor readings) and nodes that are interested in them (determined by filter conditions) caches them locally. E.g. node A can regard B as a neighbor but not vice versa; A finds node Bs sensor readings interesting and so caches what B shares. Broadcast is used for both data sharing and neighbor discovery. The authors state that the broadcasts must be cheap in terms of power consumption.

Abstract Regions Abstract regions [55] is a set of programming abstractions that are similar to Hood, but adds mechanisms for data aggregation and controlling the resource/performance trade-off. There are four categories of abstractions in Abstract regions: neighbor discovery, addressing (nodes in the regions), data sharing and data reduction (of the shared data by some rule). The neighbor discovery may use broadcasts and is a continuous process. The data sharing can also be implemented with broadcasts, so that nodes could overhear and cache data locally. Abstract regions are defined by a rule, such as based on radio metrics or geographic data, and a node can be a member in several regions at the same time.

6.4 Duty Cycling MAC Protocols

X-MAC In X-MAC [21], nodes periodically wake up and listens. The way broadcasting should be performed is not specified in the X-MAC paper, but one way is transmitting a strobe train for a period longer than the sleeping period before transmitting the actual data packet. The number of strobes is controlled by the parameters strobe train length and strobe wait time. Networks using X-MAC have scaling problems due to the congestion that may occur if many nodes are colocated.

ContikiMAC ContikiMAC is similar to X-MAC in that it is an LPL scheme which periodically wakes up to listen for traffic, but uses CCA instead of idle listening. For every listening period, it samples twice with CCA with a small

sleeping delay in between. This makes for an extremely low radio utilization. Broadcasting in Contiki-MAC is similar to X-MAC, but instead of transmitting strobes, the data packet is transmitted repeatedly so that when a receiver wakes up, it receives the data packet immediately and can go back to sleep. The problems are similar to X-MAC, as congestion and biased effort are still issues. Latency is lower for the receiver, but as there are still more traffic, this limits the possibility for quick responses until the entire broadcast is over.

RI-MAC RI-MAC [49] is an LPP scheme. Nodes waking up transmit a beacon and listens for a short while before going back to sleep. A node wanting to send must await the probe and then transmit. Broadcasts are performed by listening for a whole sleeping period and repeatedly responding to beacons from waking nodes. Congestion and scalability are not so much a problem as in X-MAC, but disinterest and biased effort are still problems. In very dense networks that perform broadcasts often, congestion might be a problem though.

S-MAC S-MAC [60], which is a synchronized MAC protocol, supports broadcasts in two ways. One way is for the sender to repeatedly transmit the packet when the neighbors wake up on their respective time slot, this has the same drawbacks as for RI-MAC. The other way includes creating virtual clusters, which essentially is a group of nodes that have coordinated time slots. The drawback with this approach is the lower flexibility and shorter lifetime for the nodes in the virtual cluster, especially for nodes bordering two clusters as they must follow both schedules.

6.5 Communication Primitives

Implicit Announcements Implicit announcements (IA) [19] is a proposed mechanism for spreading data from one to many nodes that are using power saving MAC protocols. It works by piggy-backing the announcement, i.e. two byte data and and a two byte identifier, on the next outgoing transmission. It is disregarded at the MAC layer in the receiver unless explicitly waiting for them. This is good for propagating small amounts of data to neighbors, such as ETX to data sink. The small payload possible of only two bytes can be too small for some applications. Politecasts can have arbitrarily large payloads, and no limitations in itself regarding addresses etc. Also, in contrast to politecast, IA is not immediate but waits for an outbound transmission or timer expiration. Therefore, IA and politecast have in part different application domains.

802.15.4 Beacons 802.15.4 specifies a format for beacons [14] used in superframe architecture networks. The beacons are periodic and mark the beginning and end of a superframe period consisting of 16 timeslots. They are synchronous, thus requiring precise timing. The beacons are not intended for data transfer, but rather enabling new nodes to synchronize with the superframe timings. Politecast transmissions are intended for data transfers as well

as beacons. They can be periodic, but just as well non-periodic. There are no limits on packet size (except what is imposed by standards, radio hardware or underlying software architecture). Therefore, the 802.15.4 beacons are a very small and finite subset of the politecast application domain, as they can be implemented with politecasts, but not vice versa.

Multihop Broadcasting Multihop broadcasting is used for reaching every node in the network, even if the initiator cannot reach the farthest ends itself. The purpose is propagation of data, whether it be new parameters, global commands or a route discovery request. The following multihop broadcasting mechanisms rarely focuses on exactly how the broadcast transmission itself would be carried out, assuming that the MAC layer takes care of that, but rather focuses on whether or not a node should retransmit a broadcast it has received. Other mechanisms can use a counter- or probability based scheme, an additionally covered area scheme or neighbor knowledge schemes [58].

Flooding Indiscriminate flooding is the worst case of flooding mechanisms. All nodes receiving a flooding packet retransmits it if it has not already done that. This can lead to the broadcast storm problem [45]. A sequence number in the packet is used in order to stop cyclic redundant transmissions of the same packet.

Polite gossip One way of reducing the broadcast storm problem is the polite gossip mechanism in Trickle [42], in which a node which hears a broadcast does not retransmit it unless it has something new to add. Trickle uses this for propagating code over a deployed WSN.

Chapter 7

Conclusion

With this thesis, I identified four problems with broadcast, and proposed how a new network primitive, politecast, could solve them. With simulations and experiments I showed how the four problems affect node lifetime and performance in low power wireless sensor networks. I implemented politecast in the Contiki operating system and evaluated it against broadcast and showed that politecast can offer a longer lifetime and higher performance.

Using politecast requires a new way of thinking, going from a push to a implicit pull way of communication. The how and when to listen becomes important questions. I used two triggers in the evaluations: a periodic and a movement based but the choice of listen policy is still an open research question.

Politecast is not a broadcast replacement. They are qualitatively and semantically different. Politecast is first and foremost suitable when a node shares information that few are interested in, but it does not know beforehand which nodes are. It is less suitable for applications using unexpected and critical transmissions (e.g. an alarm signal).

7.1 Advantages with Politecast

Any node that is interested and listening can take part of a politecast transmission as politecast is not addressed to any recipient. This makes it good for sharing non-critical data with many.

The burden lies mostly with the receiver, which must be awake and listening for the politecast transmission. This is in contrast to broadcasting in e.g. X-MAC, ContikiMAC or RI-MAC. This is advantageous in cases where it is more important that the senders rather than the receivers have a long lifetime, such as in the Lega system.

Politecast makes for a low latency transmission as it does not have to do repeated transmissions. The enables high performance in interaction rich applications such as the Lega. It also minimizes overhearing to nearby nodes, not disturbing them if they are not interested, making more dense networks possible.

This also makes it good for applications where one or many shares information often that few are interested in, again like in the Lega system.

Politecast is more deterministic in terms of radio utilization than broadcast because of the overhearing and congestion problems. This enables designing an application to consume a pre-deployment fixed amount of power and thus predict lifetime, in contrast to broadcast in which it depends on the number of neighbors. For example, a neighbor discovery service can be designed so that it always uses 7 % radio DC.

7.2 Limitations of Politecast

Politecast requires any receiver to be explicitly listening for transmissions, making politecast unsuitable for unexpected transmissions as the sender cannot know that the receiver is awake to receive the data.

If the listening policy is poor, the lifetime will be reduced due to excessive listening. How to trigger listen and for how long to listen are open questions that need to be handled in each application. There is no generic answer but it also enables flexibility as the tradeoff between performance and lifetime is exposed.

7.3 Future Work

Listen Policy For a continuous service, the explicit listening can amount to a high power consumption if not implemented carefully, and thus a proper listening policy must be used. The question is when and for how long to listen. It could for example be triggered by sensor readings, e.g. movement measured with an accelerometer, or a periodic timer trigger. The authors of Disco [28] created a mechanism for deterministic coexisting time slots, with no synchronization messages being transmitted but based on pairs of prime numbers. This is different from synchronous MAC protocols, as they need synchronization messages to work.

Just as important as when to listen is for how long, or when to stop listening. It could be for a fixed amount of time or until an event occurs. Stopping when the node receives the first packet (or a specified number of packets, or from a specified receiver, or of a specified type) could form the base of a content aware network. For example, say that node A is interested in the temperature and the mobility of the LR-WPAN. Then it would wait for a temperature sensor message and an accelerometer sensor message.

I am convinced that there is no generic solution suitable for all applications, but it is still an interesting question. By evaluating and trying out different policies in different applications, perhaps a cartotheque of suggested listening policies can be developed that can aid an application designer.

Figure 7.1: Instead of listening with 100 % duty cycle (left) as it is in the current implementation of politecast, duty cycling the listening and sending the packet twice within a time shorter than the sleep period (right) will lower the total utilization considerably. The duty cycles before could now be $DC_S = 1\%$ and $DC_{R_1} = 100\%$ and in the enhanced version $DC_S = 2\%$ and $DC_{R_1} = 50\%$.

Duty Cycled Listening A high amount of listening will reduce node lifetime, but using similar techniques to LPL to duty cycle the listening, the cost can be spread out and on a network level, lesser. If the listening is duty cycled at 50 % and every politecast is sent twice with a delay, this will still ensure reception. As politecast is deterministic (i.e. predictable radio duty cycle as seen in Figure 5.18and 5.21) this would make it possible to divide the burden between sender and receiver and pre-deployment know the resulting radio utilization for that service. Figure 7.1 shows how the example with 50 % will work, going from a collective utilization of 50.5 % to 26 %.

This would give the application designer the choice of which node will bear the largest burden. E.g. node type A is more likely to use the sensors less and can take a larger transmission burden than receiver node type B. An offline burden divider optimizer could, based on a few inputs, find an optimum setting for politecast and the listen policy.

Other MAC protocols Politecast has so far been implemented and evaluated in ContikiMAC and X-MAC, both asynchronous LPL protocols doing broadcast in a similar way. Politecast should be evaluated with more different protocols, such as an LPP protocol.

7.4 Discussion

One must always question the validity of the tools and results.

Goodness of the COOJA Network Simulator As measurements from the COOJA simulator was used in the evaluations, the accuracy of the simulator and radio model must be questioned. In Figure 5.24 (page 47) I examined the effects of congestion with an experiment. I recreated the same evaluation in the COOJA simulator, placing two nodes at the same distance as I did in reality, and compared the measurements between the two cases as seen in Figure 7.2.

From this very small comparison, it is impossible to reach any generic conclusions about COOJA accuracy, but it hints that COOJA with the radio model used is less accurate on the effects on radio congestion and interference. With no congestion, COOJA is accurate; the corresponding radio duty cycles are off by

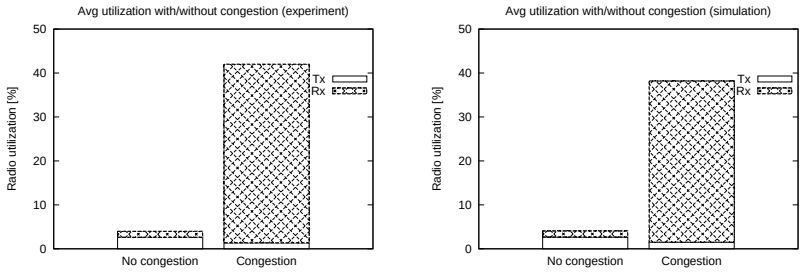


Figure 7.2: The accuracy of COOJA is hinted by the comparison of these two graphs where the left is data from experiments and the right is from simulations. COOJA seems to be slightly conservative as in the effects of congestion not being as severe as in experiments.

0.2 %, ± 0.0 % and 1.9 % but with congestion, the simulation underestimated the effects by 5.7 %, 9.8 % and 11.2 %, all the better (more optimistic) than the experiment. Also, the number of successfully received beacons per hour were overestimated by 8.0 %. The inaccuracies can be related to the use of UDGM — a recognized to be inaccurate model [38] of radio transmissions, but it hints that the results for the broadcast implementation may not perform as well in real life conditions as in the simulations in the evaluation.

Generalizability The politecast communication primitive was implemented with the C programming language in the Contiki operating system, evaluated on Sentilla JCreate, Tmote Sky and in a simulator. However, the principle behind politecast is not bound to any OS or programming language.

7.5 Conclusion

With this thesis, I identified four problem areas with broadcast: congestion, biased effort, overhearing and latency. I implemented and evaluated politecast, a new communication primitive for one-to-many communication. In the evaluation, I showed that politecast has a low, fixed and predictable communication cost and is particularly well suited in applications where the power consumption of the sender is of higher concern than the receivers, or where communication characteristics are of a "many transmit often, few are seldom interested". It is not a broadcast replacement as it is ill-suited when a transmission is unexpected, but it is another tool in the application designers toolbox.

Appendix A

API Reference

Open a politecast connection. Arguments: pointer to a politecast struct, channel number and pointer to a struct with politecast callback function pointers.

```
1 void politecast_open(struct politecast_conn *c,
2       uint16_t channel, const struct politecast_callbacks *u);
```

Listing A.1: Politecast open connection

Close a politecast connection. Arguments: pointer to the politecast struct.

```
1 void politecast_close(struct politecast_conn *c);
```

Listing A.2: Politecast close connection

Transmit the contents in packet buffer. Arguments: pointer to a politecast struct. Returns the result from the Rime send function;

```
1 uint16_t politecast_send(struct politecast_conn *c);
```

Listing A.3: Politecast send

Start politecast listen. Arguments: pointer to a politecast struct and the time to listen, measured in clock ticks. Invokes timeout callback when time is out.

```
1 void politecast_listen(struct politecast_conn *c,
2          clock_time_t listen_time);
```

Listing A.4: Start politecast listen mode

Prematurely stop the listening. Does not invoke timeout callback. Arguments: pointer to a politecast struct.

```
1 void politecast_listencancel(struct politecast_conn *c);
```

Listing A.5: Politecast stop listening

Check if we are currently listening. Arguments: pointer to a politecast struct. Returns 0 if not, 1 if we are listening.

```
uint16_t politecast_islistening(struct politecast_conn *c);
```

Listing A.6: Politecast is listening

Set the transmission power this connection shall use. Arguments: pointer to a politecast struct, the transmission power. Returns 0 if requested transmission power is out of possible range, 1 elsewise.

```
uint16_t politecast_set_txp(struct politecast_conn *c,
         char txp);
```

Listing A.7: Politecast set txp

Get the transmission power this connection uses. Arguments: pointer to a politecast struct. Returns the transmission power.

```
uint16_t politecast_get_txp(struct politecast_conn *c);
```

Listing A.8: Politecast get txp

Set the Receiving callback that is invoked when the connection receives a politecast packet on the channel. Arguments: pointer to a politecast struct and pointer to a Rime address struct, which will be set to the address of the sender.

```
void (* recv)(struct politecast_conn *c, rimeaddr_t *from);
```

Listing A.9: Politecast receive callback

Set the Timeout callback that is invoked when a politecast listen ends due to timer expiring. Arguments: pointer to a politecast struct, which listen timer expired.

```
void (* timeout)(struct politecast_conn *c);
```

Listing A.10: Politecast timeout callback

Two structures define a politecast connection: the politecast connection struct, and the callback struct with function pointers.

```
/* A politecast connection struct */
struct politecast_conn {
  struct broadcast_conn c;
  const struct politecast_callbacks *cb;
  struct ctimer listen_timer;
  char txp; /* Transmission power */
};
/* Callbacks for a politecast connection */
struct politecast_callbacks {
  void (* recv)(struct politecast_conn *c, rimeaddr_t *from);
  void (* timeout)(struct politecast_conn *c);
};
```

Listing A.11: Politecast structures

Appendix B

Code Examples

B.1 Setting Up a Politecast Connection

```
#define BEACON_TXP 15;
#define BEACON_CHANNEL 1337;
static struct BEACON_PACKET bp;
static struct politecast_callbacks p_conn_cb = {NULL, NULL};
static struct etimer et;
bp.txp = BEACON_TXP;
politecast_open(&p_conn, BEACON_CHANNEL, &p_conn_cb);
politecast_set_txp(BEACON_TXP);
while(1){
  bp.val = some_function();
  packetbuf_clear();
  packetbuf_copyfrom(&bp, sizeof(struct BEACON_PACKET));
  packetbuf_set_datalen(sizeof(struct BEACON_PACKET));
  politecast_send();
  etimer_set(&et, CLOCK_SECOND);
  PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
}
```

Listing B.1: Setting up a politecast connection.

B.2 Beacon Node

This example opens a politecast connection and periodically transmits a beacon packet.

```
#define BEACON_TXP 15;
#define BEACON_CHANNEL 1337;
PROCESS (autostart_pr, "Autostart proc");
static void
rcv(struct politecast_conn *p, const rimeaddr_t *from){
  /* Receive callback */
  printf("Rcv from %u.%u\n", from->u8[0], from->u8[1]);
}
```

```
9  static struct BEACON_PACKET{
10   uint8_t txp;
11   uint16_t val;
12 }bp;
13 static struct politecast_callbacks p_conn_cb = {rcv, NULL};
14 static struct etimer et;
15
16 PROCESS_THREAD(&autostart_pr, ev, data){
17   PROCESS_EXITHANDLER(politecast_close(&p_conn););
18   PROCESS_BEGIN();
19   bp.txp = BEACON_TXP;
20   politecast_open(&p_conn, BEACON_CHANNEL, &p_conn_callbacks);
21   politecast_set_txp(BEACON_TXP);
22   while(1){
23     bp.val = get_number_of_neighbors();
24     packetbuf_clear();
25     packetbuf_copyfrom(&bp, sizeof(struct BEACON_PACKET));
26     packetbuf_set_datalen(sizeof(struct BEACON_PACKET));
27     politecast_send();
28     etimer_set(&et, CLOCK_SECOND);
29     PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
30   }
31   PROCESS_END();
32 }
```

Listing B.2: Politecast beacon node example.

B.3 Listen Mode Triggering

Here, the listening is triggered by movement, as detected by an accelerometer.

```
1 if(!politecast_islistening(&p_conn)) {
2   if (accmeter.x >= THRESH_HI || accmeter.x <= THRESH_LO){
3     politecast_listen(&p_conn, CLOCK_SECOND * 10);
4   }
5 }
```

Listing B.3: Politecast listen triggered by accelerometer

Appendix C

Developing Environment

Software for the nodes is written in the C programming language. It is compiled in a terminal with the GCC compiler toolchain and either uploaded to the node using a boot strap loader (BSL) or run in COOJA simulations. For simulations, position data is generated with software written in Python. Python is also used for measurement data parsing and compilation.

Instant Contiki, Ubuntu A virtual machine image of Ubuntu 8.04 with all the necessary files for software development in Contiki can be downloaded on the Contiki homepage [12]. It is called Instant Contiki. The Contiki version used was version 2.3 with CVS updates until 2010 april 10.

Compiler — GCC GNU Compiler Collection is a set of free tools for compiling and linking. The target platforms include IA-32 (x86), Atmel AVR and Texas Instruments MSP430. The support for C99 extends to some but not all features. The set used for this thesis is called MSPGCC [7].

Contiki Simulator — COOJA For simulating nodes with Contiki operating system, the simulator COOJA is used. COOJA is written in Java and can simulate MSP430-based (among others) nodes and gives full insight to variables, registers and source code execution. It is developed at SICS and uses the MSPsim module for simulating the microcontroller.

MSP430 Simulator — MSPsim MSPsim is a Java-based Texas Instruments microcontroller simulator developed at SICS [9].

Portable Python Portable Python v1.1 [10] is a distribution of a Python 2.5.4 interpreter. The distribution also contains the Numeric package, which is used for handling array structures.

Appendix D

Glossary

ACK Acknowledgment (packet).

BCN, Beacon general term for a small "Hello"-like transmission. Often periodic. Used in neighbor discovery and other applications.

CCA Clear Channel Assessment.

CSMA-CA Carrier Sense, Multiple Access – Collision Avoidance; before transmitting, the radio transceiver checks the media for radio energy. If detected by CCA, it waits a specified or random time (the backoff window) before trying again.

dBm dBm is the power ratio in decibels (dB) normalized to zero dBm being equal to one milliwatt (mW). Every increase or decrease with 3 dBm means doubling or halving the power.

DC Duty Cycle. For example, the duty cycle for radio transmitting is $\frac{\Sigma(time\ in\ transmitting\ mode)}{total\ time}$.

DSSS Direct-Sequence Spread Spectrum, each of the transmitted data bits are encoded into several transmission bits that are modulated on a 2.4 GHz carrier wave. At the receiver the signal is demodulated and the bits are decoded to data bits. The carrier wave makes antennas with cm-size possible (instead of km-size) and the encoding makes the signal more robust to interference [47].

ETX Estimated number of Transmissions (to e.g. specific neighbor, data sink). A metric often used in routing tables etc for finding the "shortest" route.

HAL Hardware Abstraction Layer, used for separating low level device programming from higher level application programming.

IEEE Institute of Electrical and Electronics Engineers.

ISM Industrial, Scientific and Medical.

LPL Low-Power Listening, a MAC scheme in which the nodes wakes up and listens for a while for traffic before going back to sleep. Example: X-MAC.

LPP Low-Power Probing, a MAC scheme in which the node wakes up and probes the others for data by sending a beacon, which nodes can respond to. Example: RI-MAC.

LQI Link Quality Indicator, a metric describing the amount of bit errors encountered during reception of a packet.

LR-WPAN Low Rate Wireless Personal Area Network.

Mote A wireless sensor device used in WSNs. Also called *node*.

Node Generic term for an electronic device attached to a network. In this thesis, it is used meaning a wireless sensor node in WSNs. Also called *mote*.

NULLMAC a MAC scheme that does no power saving efforts. The radio is always on.

RSSI Received Signal Strength Indicator, indicates the signal strength in dBm.

RTS, CTS Request To Send, Clear To Send.

RWMM Random Waypoint Mobility Model.

Strobe a small packet only consisting of the minimal 802.15.4 header. Used for waking up sleeping nodes for receiving data. The strobe train is called *preamble*.

TDMA Time Division, Multiple Access. Time is divided into slots and assigned to nodes.

UDGM Unit Disk Graph Model, a model for radio transmissions.

References

[1] Datasheet for Antenova Impexa chip antenna. http://www.antenova.com/?id=875. Retrieved 2009-10-05.

[2] Datasheet for Freescale MMA7260 accelerometer. http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MMA7260QT. Retrieved 2009-10-05.

[3] Datasheet for ST Microelectronics M25P80 flash memory. http://www.st.com/stonline/products/literature/ds/8495/m25p80.pdf. Retrieved 2009-10-05.

[4] Datasheet for Texas Instruments CC2420 radio transceiver. http://focus.ti.com/docs/prod/folders/print/cc2420.html. Retrieved 2009-10-05.

[5] Datasheet for Texas Instruments MSP430F1611 microcontroller. http://focus.ti.com/docs/prod/folders/print/msp430f1611.html. Retrieved 2009-10-05.

[6] Design note 007 - 2.4 ghz Inverted F Antenna (Rev. B) 2008-apr-04. http://www.ti.com. Retrieved 2010-03-30.

[7] Home page of GCC compiler. http://mspgcc.sourceforge.net/. Retrieved 2009-10-01.

[8] Home page of Liljevalchs art exhibition. http://www.liljevalchs.stockholm.se/. Retrieved 2009-05-01.

[9] Home page of MSPsim, the MSP430-family simulator. http://www.sics.se/project/mspsim/. Retrieved 2009-10-01.

[10] Home page of Portable Python. http://www.portablepython.com/. Retrieved 2009-10-01.

[11] Home page of Sentilla corporation. http://www.sentilla.com/store/. Retrieved 2009-05-01.

[12] Home page of the Contiki operating system. http://www.sics.se/contiki/. Retrieved 2009-10-20.

[13] HP foresees billions of sensor nodes. http://www.readwriteweb.com/archives/cense_hp_labs.php. Retrieved 2009-10-05.

[14] IEEE 802.15.4a-2007 standard. http://standards.ieee.org/getieee802/download/802.15.4a-2007.pdf. Retrieved 2009-07-01.

[15] Informationweek Global CIO article on WSN emerging market. http://www.informationweek.com/news/global-cio/showArticle.jhtml?articleID=57702816. Retrieved 2009-10-05.

[16] Wikipedia entry on solar panels. http://en.wikipedia.org/wiki/Solar_cell. Retrieved 2010-03-30.

[17] Wikipedia entry on WSNs. `http://en.wikipedia.org/wiki/Wireless_sensor_network`. Retrieved 2010-03-30.

[18] A. Dunkels and B. Grönvall and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[19] A. Dunkels and L. Mottola and N. Tsiftes and F. Österlind and J. Eriksson and N. Finne. Implicit Announcements: Rethinking the use of broadcast in mobile sensor networks. Technical Report T2009:09, Swedish Institute of Computer Science.

[20] Luis Bernardo, Rodolfo Oliveira, Miguel Pereira, Mario Macedo, and Paulo Pinto. A wireless sensor MAC protocol for bursty data traffic. 2007.

[21] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, New York, NY, USA, 2006. ACM.

[22] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[23] A. Dunkels. uIP - a TCP/IP stack for 8- and 16-bit microcontrollers. `http://www.sics.se/~adam/uip/`. Retrieved 2009-10-01.

[24] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[25] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, 2006.

[26] Adam Dunkels, Fredrik Österlind, and Zhitao He. An adaptive communication architecture for wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 335–349, New York, NY, USA, 2007. ACM.

[27] Adam Dunkels, Fredrik Österlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32, New York, NY, USA, 2007. ACM.

[28] Prabal Dutta and David Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 71–84, New York, NY, USA, 2008. ACM.

[29] Prabal Dutta and David Culler. Mobility changes everything in low-power wireless sensornets. In *Proceedings of the 12th Workshop on Hot Topics in Operating Systems (HotOS-XII)*, Monte Verita, Switzerland, May 2009.

[30] Prabal Dutta, Jay Taneja, Jaein Jeong, Xiaofan Jiang, and David Culler. A building block approach to sensornet systems. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 267–280, New York, NY, USA, 2008. ACM.

[31] A. El-Hoiydi and J. D. Decotignie. WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks. volume 1, pages 244–251 Vol.1, 2004.

[32] Laura Marie Feeney. Exploring semantic interference in heterogeneous sensor networks. In *HeterSanet '08: Proceeding of the 1st ACM international workshop on Heterogeneous sensor and actor networks*, pages 45–52, New York, NY, USA, 2008. ACM.

[33] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report, UCLA Computer Science Department, 2002.

[34] Katherine Isbister and Kristina Höök. On being supple: in search of rigor without rigidity in meeting new design and evaluation challenges for HCI practitioners. In *CHI*, pages 2233–2242, 2009.

[35] Xiaofan Jiang, Prabal Dutta, David Culler, and Ion Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 186–195, New York, NY, USA, 2007. ACM.

[36] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet, 2002.

[37] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[38] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yougu Yuan, and Chip Elliott. Experimental evaluation of wireless simulation assumptions. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 78–82, New York, NY, USA, 2004. ACM Press.

[39] Branislav Kusy, Akos Ledeczi, and Xenofon Koutsoukos. Tracking mobile nodes using RF Doppler shifts. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 29–42, New York, NY, USA, 2007. ACM.

[40] J. Laaksolahti, J. Tholander, M. Lundén, J. Solsona Belenguer, A. Karlsson, and T. Jaensson. The Lega: A device for leaving and finding tactile traces. In *Fifth International Conference for Tangible, Embedded and Embodied Interaction*. ACM, 2011.

[41] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: an operating system for sensor networks. In *in Ambient Intelligence*, 2004.

[42] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of NSDI'04*, March 2004.

[43] Konrad Lorincz and Matt Welsh. Motetrack: a robust, decentralized approach to RF-based location tracking. *Personal and Ubiquitous Computing*.

[44] Luca Mottola and Gian Pietro Picco. Programming wireless sensor networks with logical neighborhoods. In *InterSense '06: Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, page 8, New York, NY, USA, 2006. ACM.

[45] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.

[46] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, November 2006.

[47] A. Gjelstrup S. Jensen and V. Berti. *Datakommunikation*. Liber AB, 2000.

[48] Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. The beta-factor: measuring wireless link burstiness. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 29–42, New York, NY, USA, 2008. ACM.

[49] Yanjun Sun, Omer Gurewitz, and David B. Johnson. RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 1–14, New York, NY, USA, 2008. ACM.

[50] Petra Sundström, Tove Jaensson, Kristina Höök, and Alina Pommeranz. Probing the potential of non-verbal group communication. In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 351–360, New York, NY, USA, 2009. ACM.

[51] Petra Sundström, Anna Staahl, and Kristina Höök. In situ informants exploring an emotional mobile messaging system in their everyday practice. *Int. J. Hum.-Comput. Stud.*, 65(4):388–403, April 2007.

[52] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM.

[53] Thiemo Voigt, Adam Dunkels, Joakim Eriksson, and Niclas Finne. Experiences from two sensor network deployments: self-monitoring and self-configuration keys to success. In *Proceedings of Wired/Wireless Internet Communications: 6th International Conference, WWIC 2008: Proceedings*, page 12, Tampere, Finland, 2008. Lecture notes in computer science; 5031. DOI: 10.1007/978-3-540-68807-5.

[54] Benjamin W. Wah. *Wiley Encyclopedia Of Computer Science And Engineering*. John Wiley & Sons, 2007.

[55] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proc. USENIX/ACM NSDI'04*, San Francisco, CA,, March 2004.

[56] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381–396, Berkeley, CA, USA, 2006. USENIX Association.

[57] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: a neighborhood abstraction for sensor networks. In *Proc. ACM MobiSys'04*, Boston, MA, USA, June 2004.

[58] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205, New York, NY, USA, 2002. ACM.

[59] Joey Wilson and Neal Patwari. Through-wall tracking using variance-based radio tomography networks. *CoRR*, abs/0909.5417, 2009.

[60] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.